£1.00

# ELBUG FOR THE ELECTRON

Vol 1 No 9 Aug/Sept 1984

## Kayak Kapers

### Games

* Codebreaker
* Dartboard

Plus

* Harmonograph
* Mini Text Editor
* Build a House

Plus

* Monitors Reviewed
* Latest Games Reviewed
* Hints and Tips
* And much more

# EDITORIAL

This Month's Magazine

This issue of ELBUG covers the two months of August and September, so the next issue will be that for October. The contents of this issue have a pronounced visual flavour to them. Apart from the three games, all of which provide excellent screen displays, there are two further programs of a highly graphical nature. 'Harmonograph' will produce endless patterns after the style of Spirograph, giving many hours of fascinating pleasure, while 'Build a House' will probably please our younger readers in the way in which the colourful picture of a house and garden is built up on the screen. Watch out here for the cat.

Our series on Electron graphics is now drawing to a close, and this month we show how to extend the choice of colours available, by producing all manner of striped colours and shades to enhance you own graphics displays.

Now that interfaces for joy-sticks and printers are becoming quite widely available, we have included a most useful and practical application in the form of a text editor. This allows you to create and edit text for all sorts of purposes and even if you don't yet have a printer with your Electron, may still prove of great value in communicating with other ELBUG members and sending us letters and articles in the ELBUG office.

THE EXPANDING ELECTRON

Acorn tell us that they are now sending out over 20,000 Electrons a month and that this is rapidly becoming one of the best selling micros in its price range. Many other suppliers are also extending the range of add-ons for the Electron, evidenced by several of the stands at the recent Micro User Show at Alexandra Palace. We shall be looking at all the new developments for the Electron and expect to carry reviews of the latest products in the next issue.

Mike Williams

# NOTICE BOARD NOTICE BOARD NOTICE BOARD NOTICE BO

Magazine Cassette

All the programs in this month's magazine are available on cassette together with an additional program by S.Fox which provides a test card on your TV or monitor screen to go with our review of monitors in this issue. See the back cover for full ordering details.

MAGAZINE BINDERS

Several readers have written to us about binders for their copies of ELBUG, and we are pleased to announce that the details of this have now been agreed. See the inside back cover for further information.

Hint Winners

This month we have awarded three £5 prizes for the best hints, and the winners are D.Robinson, G.Edwards and N.kelly. Any new hints will always be most welcome.

Flowers of Hell

This program, published in our last issue, was unfortunately subject to error during production which resulted in one line being omitted from the listing. This relates to line 1770 and the correct listing at this point should read as follows:

```
1760 VDU23,227,255,255,255,127,63,3,3,1
1770 VDU23,228,24,126,126,255,255,126,126,24
```

We are sorry for this omission.

# ELBUG MAGAZINE

## GENERAL CONTENTS


2 Editorial
4 Codebreaker
7 Harmonograph
9 Mini Text Editor
13 Kayak Kapers
15 Build a House
18 Postbag
19 Monitors for the Electron
23 Electron Graphics (Part 9)
26 More Games Reviewed
27 More New Add-ons for the Electron
28 Debugging Programs (Part 2)
31 Dartboard
33 Points Arising


### HINTS, TIPS AND INFO


6 Function Key Listing of Envelopes
6 LISTO Abbreviation
6 Position of System Variables in Memory
8 Disabling the Keyboard
12 REM Statements in DATA Statements
12 BREAK from within Basic
22 Mode Selection on Break
25 Top Line of Text
25 Shaded Lettering for Titles
30 Rounding Decimals
30 Mode Change by VDU Command
30 Caps Lock Poke
30 Spooling Graphics


### PROGRAMS


4 Codebreaker Game
7 Harmonograph Display
9 Mini Text Editor
13 Kayak Kapers Game
15 Build a House
23 Graphics Demonstration
31 Dartboard Game

# CODEBREAKER

**By G. Huszcza**

This game makes good use of the range of colours available in mode 2 to provide an excellent implementation of the game popularly known as 'Mastermind', in which you have to break a secret code, based on a sequence of colours.

The program Codebreaker is comparatively short and should be typed into your Electron and saved on cassette before starting to play. In the game the computer will secretly set up a code of 4 coloured pegs out of a total of 6 colours available. You then have a maximum of 10 chances to guess the correct sequence and win.

The six colours are numbered from 1 to 6, and you enter your sequence by just pressing the four corresponding number keys (those numbered 1 to 6 of course) and confirm your guess by pressing Return. Until you press Return you can delete any or all of the colours already selected by using the Delete key. As the colours are selected they are displayed clearly on the left hand side of the screen.

Once you have completed your latest try the computer will tell you how close you are by displaying the number of 'cows' and 'bulls' you have scored. A 'cow' - shown by a white marker or peg - indicates the correct colour but in the wrong place, and a 'bull' - shown by a black peg - indicates the right colour and in the right place. Clearly the code is guessed correctly when you score four 'bulls' (black markers).

When you are playing, remember that colours may be repeated, both in the secret code and in your guess. It will also pay you to use your early guesses to try and gather useful information, rather than guessing the correct code outright.

This is a fascinating and demanding game which will keep you entertained for hours, particularly with the colourful version provided by this program.

PROGRAM NOTES

The principal procedures used in this program are as follows:

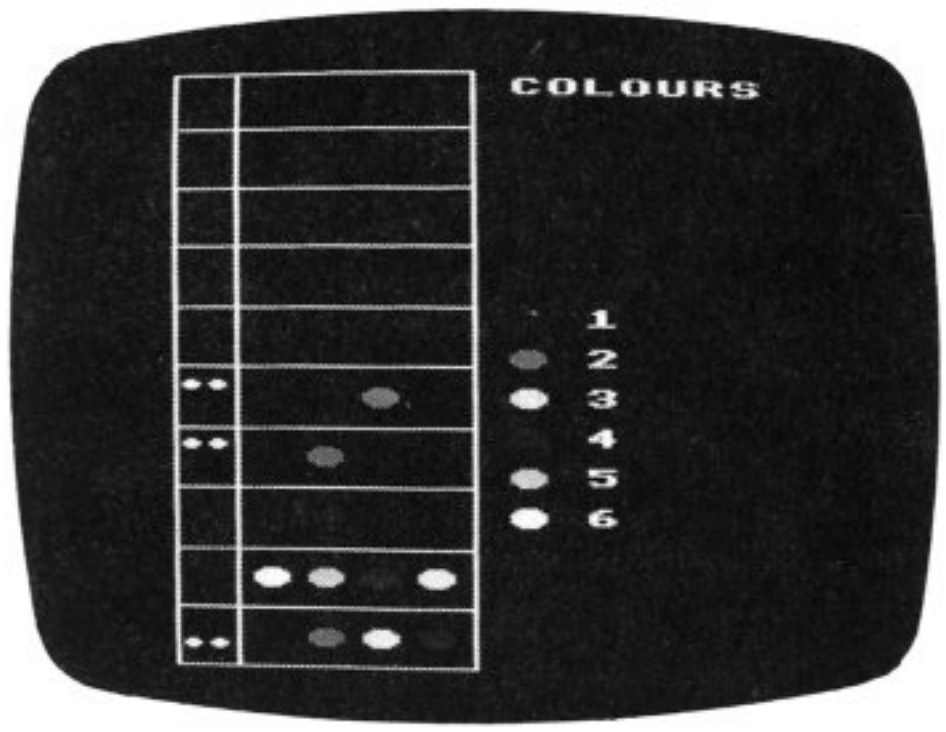
| | |
|---|---|
| title | Displays main title page. |
| init | Initialises variables, user defined characters and sound envelopes. |
| drawgrid | Draws the outline of the game display. |
| startgame | Randomly selects the secret code. |
| choose-colours | Deals with the selection of colours by the player. |
| check-colours | Checks the latest guess. |
| gameend | Displays the score etc at the end of a game. |

The two procedures, PROCchoosecolours and PROCcheckcolours, also call some further short procedures whose functions should be obvious from their names. They deal with the display and removal of coloured pegs as guessed by the player, and the display of the 'cows' and 'bulls'.

```
10 REM PROGRAM CODEBREAK
20 REM VERSION E0.2
30 REM AUTHOR  G.HUSZCZA
40 REM ELBUG   AUG/SEP.84
50 REM PROGRAM SUBJECT TO COPYRIGHT
60 :
100 ON ERROR GOTO1970
110 DIM YOURPEG%(4),PEG%(4),TEMP%(4)
120 MODE1
130 PROCtitle
140 REPEAT
150 MODE2
160 VDU23,1,0;0;0;0;
170 VDU24,48;22;576;1001;
180 PROCinit
190 PROCdrawgrid:PROCstartgame
200 REPEAT
210 PROCchoosecolours
220 PROCcheckcolours
230 UNTIL TRYS%=10 OR FOUND=TRUE
240 PROCgameend
250 UNTILNEWGAME=FALSE
260 VDU23,1,1;0;0;0;
270 MODE6
```

→

```
 280 END
 290 :
 1000 DEFPROCtitle
 1010 VDU19,0,4,0,0,0:PRINTTAB(15,2);"C
OLOURS":PRINTTAB(18,4);"By";TAB(14,6);"
G.Huszcza"
 1020 PRINTTAB(4,10);"KEYS ARE AS FOLLO
WS :-";TAB(6,14);"1-6 FOR COLOURS";TAB(
6,16);"DELETE TO RUB OUT";TAB(6,18);"RE
TURN TO FINISH TURN"
 1030 PRINT''TAB(4)"Black pegs signify
a correct colour in the correct hole,
and  a  white peg signifies a correct c
olour in the wrong hole."
 1040 PRINTTAB(13,30);"PRESS ANY KEY":A
%=GET
 1050 ENDPROC
 1060 :
 1070 DEFPROCinit
 1080 TRYS%=0:FOUND=FALSE:NEWGAME=TRUE:
PEGX%=192:PEGY%=87:MARKERX%=48:MARKERY%
=75:BLACKMARKS%=0:THREE=FALSE
 1090 VDU23,255,60,126,255,255,255,255,
126,60:VDU23,254,0,0,24,60,60,24,0,0
 1100 ENVELOPE2,2,6,0,0,255,0,0,126,0,0
,-126,126,126
 1110 ENDPROC
 1120 :
 1130 DEFPROCdrawgrid
 1140 GCOL0,132:CLG:GCOL0,7
 1150 MOVE48,22:DRAW48,1001:DRAW576,100
1:DRAW576,22:DRAW48,22
 1160 FORF%=120TO904STEP98:MOVE48,F%:DR
AW576,F%:NEXT
 1170 MOVE160,22:DRAW160,1001
 1180 GCOL0,0:VDU5:FORF%=87TO969STEP98:
FORP%=192TO480STEP96:MOVEP%,F%:VDU255:N
EXT,
 1190 VDU4:COLOUR3:PRINTTAB(10,1)"COLOU
RS"
 1200 ENDPROC
 1210 :
 1220 DEFPROCstartgame
 1230 FORF%=1TO4
 1240 PEG%(F%)=RND(7):IFPEG%(F%)=4 GOTO
1240
 1250 NEXT
 1260 COLOUR7:FORF%=2TO12STEP2:color%=F
%/2:IFcolor%>=4 THEN color%=color%+1
 1270 COLOURcolor%:PRINTTAB(10,11+F%);C
HR$(255):COLOUR7:PRINTTAB(12,11+F%);F%/
2:NEXT
 1280 COLOUR1:PRINTTAB(10,3);"PICK THE"
;TAB(10,4);"COLOURS";TAB(10,5);"YOU WAN
T";TAB(10,6);"BY";TAB(10,7);"ENTERING";
TAB(10,8);"THE NO'S"
 1290 VDU5
 1300 ENDPROC
 1310 :
 1320 DEFPROCchoosecolours
```




```
 1330 F%=1:REPEAT
 1340 REPEAT
 1350 *FX15
 1360 A%=GET
 1370 UNTILA%>48 AND A%<55 OR A%=127 OR
 A%=13
 1380 SOUND1,-15,80,5
 1390 IF A%<>127 AND A%<>13 AND F%<5 TH
EN YOURPEG%(F%)=A%-48:IF YOURPEG%(F%)>=
4 THEN YOURPEG%(F%)=YOURPEG%(F%)+1
 1400 IF A%<>127 AND A%<>13 AND F%<5 TH
EN YOURCOLOUR%=YOURPEG%(F%):PROCplacepe
g(YOURCOLOUR%,PEGX%,PEGY%):PEGX%=PEGX%+
96
 1410 IF A%=127 AND PEGX%<>192 PROCremo
ve:F%=F%-1
 1420 IF A%>48 AND A%<55 AND F%<>5 THEN
 F%=F%+1
 1430 IF A%=13 AND F%=5 THEN F%=6
 1440 UNTIL F%=6
 1450 PEGY%=PEGY%+98:PEGX%=192
 1460 TRYS%=TRYS%+1
 1470 ENDPROC
 1480 :
 1490 DEFPROCremove
 1500 PEGX%=PEGX%-96:GCOL0,0:MOVE PEGX%
,PEGY%:VDU255
 1510 ENDPROC
 1520 :
 1530 DEFPROCplacepeg(COL%,X%,Y%)
 1540 GCOL0,COL%:MOVEX%,Y%:VDU255
 1550 ENDPROC
 1560 :
 1570 DEFPROCcheckcolours
 1580 FORP%=1TO4:TEMP%(P%)=PEG%(P%):NEXT
 1590 FORF%=1TO4:IFTEMP%(F%)=YOURPEG%(F
%) THEN PROCmarker(0):YOURPEG%(F%)=8:TE
MP%(F%)=8:BLACKMARKS%=BLACKMARKS%+1
 1600 NEXT
 1610 FOR PEGS%=1TO4:FINISHED=FALSE:F%=
1:REPEAT
```

→

```
 1620 IF YOURPEG%(F%)=TEMP%(PEGS%) AND
YOURPEG%(F%)<>8 AND TEMP%(PEGS%)<>8 THE
N PROCmarker(7):YOURPEG%(F%)=8:FINISHED
=TRUE
 1630 F%=F%+1
 1640 UNTIL FINISHED=TRUE OR F%>4
 1650 NEXT
 1660 IF THREE=TRUE THEN MARKERY%=MARKE
RY%+66:THREE=FALSE ELSE MARKERY%=MARKER
Y%+98
 1670 MARKERX%=48
 1680 IF BLACKMARKS%=4 THEN FOUND=TRUE
ELSE BLACKMARKS%=0
 1690 ENDPROC
 1700 :
 1710 DEFPROCmarker(MARKCOLOUR%)
 1720 GCOL0,MARKCOLOUR%:MOVEMARKERX%,MA
RKERY%:VDU254
 1730 MARKERX%=MARKERX%+48
 1740 IF MARKERX%>96 THEN MARKERX%=48:I
F (MARKERY%-75)MOD 98=0 THEN MARKERY%=M
ARKERY%+32:THREE=TRUE
 1750 SOUND1,-15,45,4
 1760 ENDPROC
 1770 :
 1780 DEFPROCgameend
 1790 VDU4:COLOUR2:PRINTTAB(10,3);"GAME
 OVER!"
 1800 FORF%=4TO24:PRINTTAB(10,F%);SPC(1
0):NEXT
 1810 COLOUR7:PRINTTAB(10,7);"COMPUTER"
;TAB(10,8);"HAD:-"
 1820 FORF%=1TO4:COLOURPEG%(F%):PRINTTA
B(10+(F%*2),10);CHR$(255):NEXT
 1830 COLOUR7:PRINTTAB(10,18);"PRESS AN
Y";TAB(13,19);"KEY"
 1840 SOUND1,2,4,50:REM FANFARE
 1850 A%=GET:CLS:COLOUR3
 1860 PRINTTAB(4,1);"PERFORMANCE":IF FO
UND=TRUE THEN PRINTTAB(0,4);"YOU FOUND
THE RIGHT"''"ANSWER IN ";TRYS%;" TURN";
:IF TRYS%<>1 THEN PRINT"S"
 1870 COLOUR5:PRINTTAB(0,10);:IF TRYS%>
=1 AND TRYS%<=3 THEN PRINT"YOU WERE LUC
KY"
 1880 IF TRYS%>3 AND TRYS%<=6 PRINT"YOU
 DID WELL"
 1890 IF TRYS%>6 AND TRYS%<=10 AND FOUN
D=TRUE PRINT"I'LL BEAT YOU NEXT"''"
   TIME"
 1900 IF FOUND=FALSE THEN PRINT"YOU LOS
T!!!"
 1910 COLOUR3:PRINTTAB(0,16);"ANOTHER G
AME (Y/N)?"
 1920 REPEAT:ANS%=GET:UNTILANS%=78 OR A
NS%=89
 1930 IF ANS%=78 THEN NEWGAME=FALSE
 1940 CLS
 1950 ENDPROC
 1960 :
 1970 ON ERROR OFF
 1980 MODE6:IF ERR=17 END
 1990 REPORT:PRINT" at line ";ERL
 2000 END
```

# HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

FUNCTION KEY LISTING OF ENVELOPES - T.J.L.Young

This will set a function key to display the parameters of any desired envelope. These are listed in the same order as in the ENVELOPE instruction, but the envelope number should supplied in response to the prompt:

```
*K.0MO.6:INP."Number of envelope";N%:@%=0:P."Envelope ";N%;:F.t%=0 TO 12:P.",";:
E%=?(&8C0+(N%-1)*16+t%):IF(t%>0 AND t%<4) OR (t%>6 AND t%<11) THEN IF E%>127 E%=
E%-1:T%=E%EOR&FF:P."-";:P.T%;:N.:P.:@%=10:EL.T%=E%:P.T%;:N.:P.:@%=10|M
```

LISTO ABBREVIATION

The User Guide lists keywords and their abbreviated forms between pages 134 and 194, except for the LISTO command for which it says there is none. However, it documents LIST as being shortened to L., so the conclusion that shortening LISTO to L.O would do the trick, is quite correct. For example L.O1 followed by L. (or LIST) will cause a program to be listed with a space inserted between the line number and the program text. (Remember to use a capital 'O' not zero here). (See also the June issue of ELBUG).

POSITION OF SYSTEM VARIABLES IN MEMORY - P.S.Ganney

The system variables @% and A% to Z% are all directly accessible at memory locations &400 to &468. As each is an integer, each has four bytes designated to it such that the value of A% is !404 etc. So it is easy to access these system variables from within machine code programs without passing parameters using the CALL instruction (see pages 138 and 216 of the User Guide).

# HARMONOGRAPH

By David C. Nichols

Your Electron is capable of some really impressive displays. The program listed here from David C. Nichols is a good example of graphics combined with some mathematical theory which produces some fascinating patterns.

The original Harmonograph is a simple mechanical device which can draw a wide variety of complex and intricate patterns. It consists of two rigid pendula that are free to swing in any direction, with the pivot point some way down from the top. A table is attached to the top of one pendulum and a lever with a pen is attached to the top of the other. A drawing is made by fixing a piece of paper to the table, resting the pen on the paper, and setting the two pendula swinging. Their independent motion produces a relative 'wobble' of the pen which makes the patterns. By varying the heights of the pendulum bobs, and the amount and direction of the pendulum swings, different types of pattern can be produced.

The program listed here is called 'Harmony', and simulates the operation of a Harmonograph, although the program ignores friction to keep the coding simple. In real life the patterns would be determined by the operator setting the pendula swinging at random. The program simulates this by setting each pendulum to an initially random position and each is endowed with a random degree of momentum. To increase the speed of execution, the program uses a 'lookup table' for the complicated trigonometric calculations. This is done at the start of a program, and values are stored in an array for later reference. It is the calculation of these values that causes the pause at the start of the program. This technique was described in Part 4 of our series on Electron graphics in ELBUG issue 4.

## USING THE PROGRAM

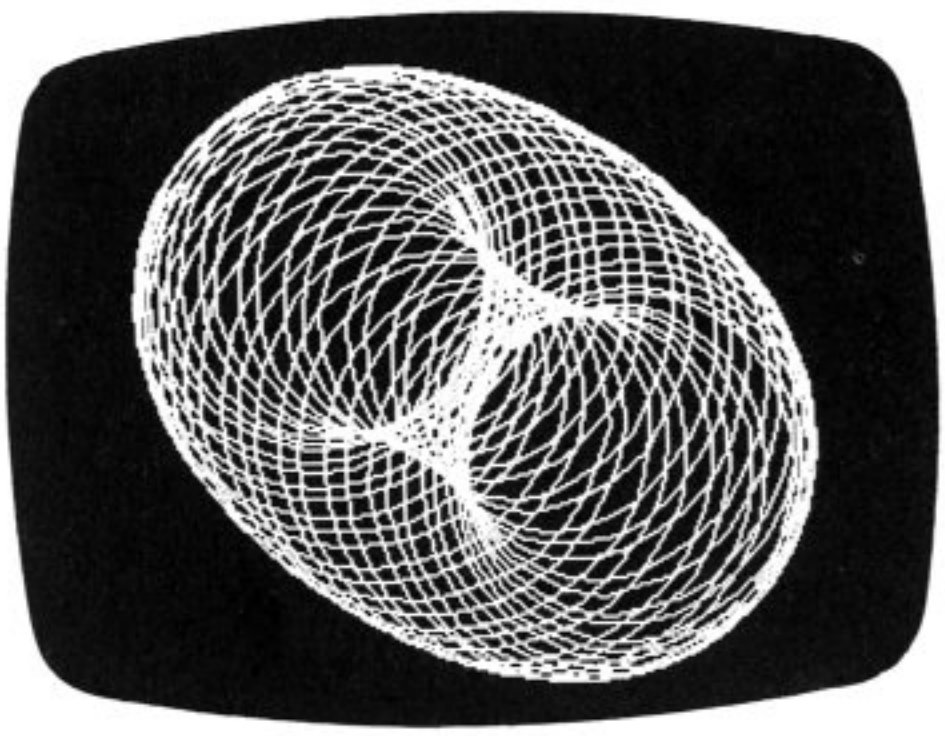
To use the program, simply type it into your Electron, save a copy on tape, and then run it. When it is in operation, pressing the Escape key will pause the program. If the space bar is now pressed, it will carry on from where it was, but if Escape is pressed again, then it will start drawing another pattern. Note that you need to press Break to leave the program after it has started drawing the patterns on the screen, as there is no exit facility built in. You may like to add one yourself, though (this could be done by testing if the Shift key is pressed at the same time that you press Escape with an INKEY -1, and performing an END if this test proved true).

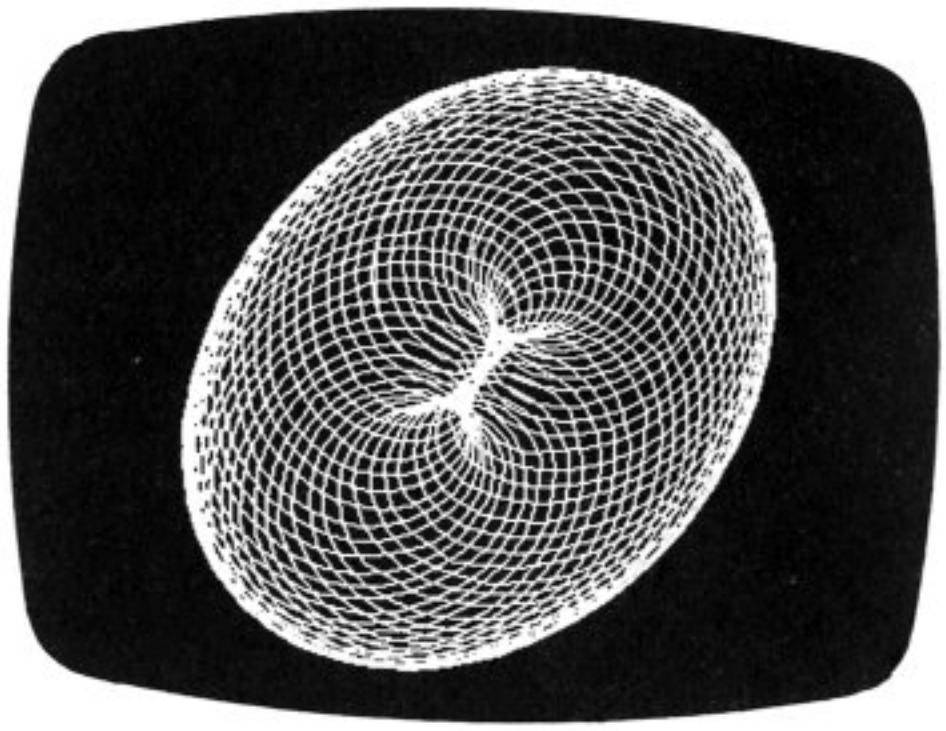
## TECHNICAL DETAILS

The program simulates the Harmonograph by considering the motion of each pendulum to be composed of two sinusoidal components at right angles. Each component has an amplitude (A1, A2, A3 and A4) and an angle of oscillation relative to some reference (R1, R2, R3 and R4). In addition each pendulum has a frequency of oscillation (F1 and F2). The relative motion of each pendulum is plotted on the screen giving a pattern similar to the so called 'Spirograph' patterns. The passage of time is simulated by I% being incremented within the REPEAT UNTIL loop at lines 350 to 410.

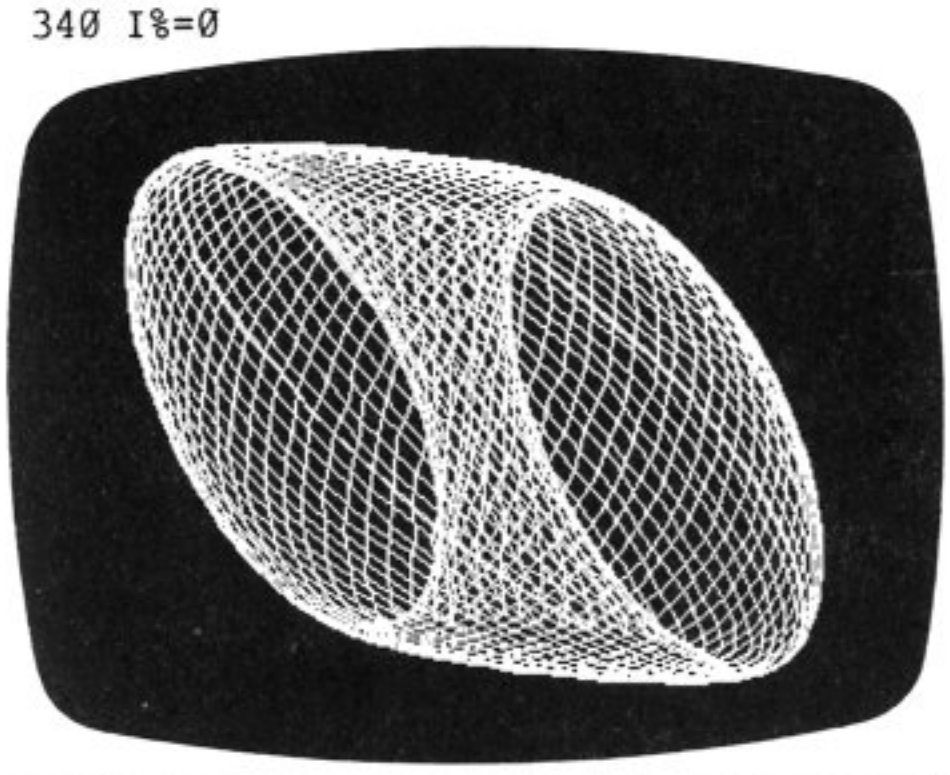
```
  10 REM PROGRAM HARMONOGRAPH
  20 REM AUTHOR DAVID C. NICHOLS
  30 REM VERSION E1.3
  40 REM ELBUG AUG/SEPT 1984
  50 REM PROGRAM SUBJECT TO COPYRIGHT
  60 :
 100 MODE 6:VDU23,1,0;0;0;0;
 110 COLOUR 129:COLOUR0
 120 PRINTTAB(13,12)"Please Wait."
 130 COLOUR 128:COLOUR1
 140 C%=255:Q%=C%/4:X%=640:Y%=512:A=400
 150 DIM sin(C%)
 160 FORI%=0TOC%
```

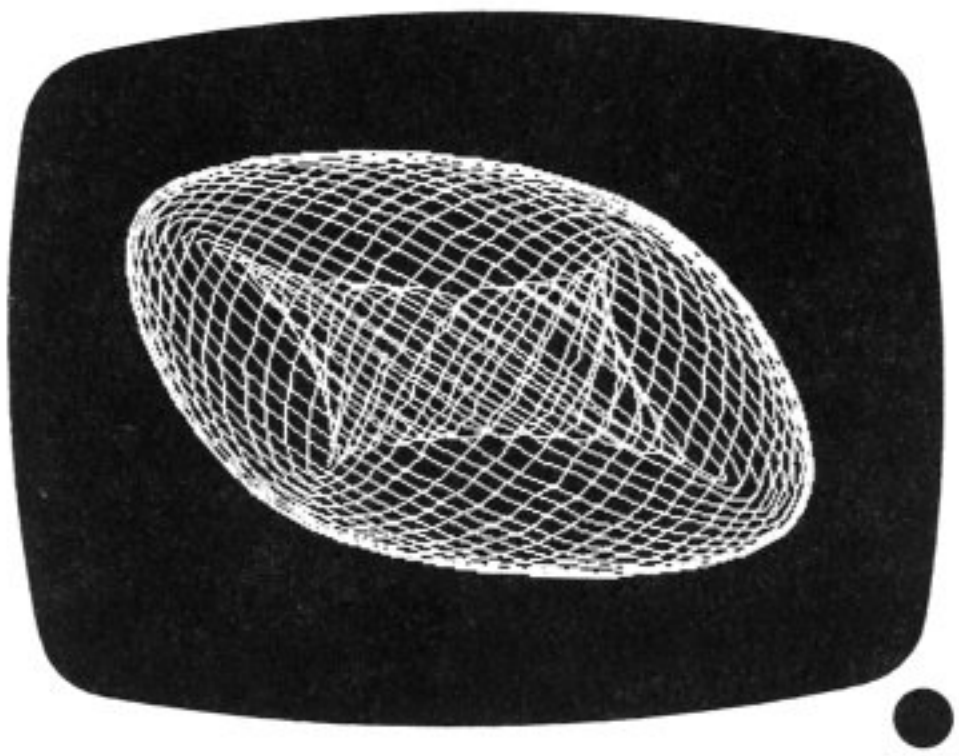
→





```
170 sin(I%)=SIN(I%*2*PI/C%)
180 NEXT
190 MODE 4:VDU19,0,4,0,0,0,0
200 ON ERROR GOTO 440
210 VDU 23,1,0;0;0;0;
220 R1=RND(C%):R2=(R1+Q%)MODC%
230 R3=RND(C%):R4=(R3+Q%)MODC%
240 A1=RND(A):A2=RND(A)
250 A3=RND(A):A4=RND(A)
260 F1=RND(1)*7:F2=RND(1)*9:B%=RND(C%)
270 X1=sin(R1)*A1:X2=sin(R2)*A2
280 X3=sin(R3)*A3:X4=sin(R4)*A4
290 Y1=sin((Q%+R1)MODC%)*A1
300 Y2=sin((Q%+R2)MODC%)*A2
310 Y3=sin((Q%+R3)MODC%)*A3
320 Y4=sin((Q%+R4)MODC%)*A4
330 M%=4
340 I%=0
350 REPEAT
360 I%=I%+1
370 T1%=F1*I%ANDC%:T2%=B%+F2*I%ANDC%
380 T3%=Q%+T1%ANDC%:T4%=Q%+T2%ANDC%
390 PLOTM%,X1*sin(T1%)+X2*sin(T3%)+X3
*sin(T2%)+X4*sin(T4%)+X%,Y1*sin(T1%)+Y2
*sin(T3%)+Y3*sin(T2%)+Y4*sin(T4%)+Y%
400 M%=5
410 UNTIL FALSE
420 END
430 :
440 IF ERR<>17 REPORT:PRINT" at line
";ERL:END
450 ON ERROR GOTO 190
460 REPEAT UNTIL GET=32
470 ON ERROR GOTO 440
480 GOTO 350
```





# HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

DISABLING THE KEYBOARD

It's possible to disable the keyboard completely with the command *FX201,1. However, as you will no longer be able to give any commands, you must be prepared to press Break. *FX201,0 does the opposite and enables the keyboard, and as these commands can be used from within Basic, it's possible to stop the user of a program from interfering with a program until required to do so.

ELBUG,
P.O.Box 5Ø,
St Albans,
Herts.
August 1984.

Dear ELBUG Reader,

Regarding your enquiry about:

# MINI TEXT EDITOR

**by David Graham**

A text editor is one of the most useful computer applications that you are likely to find for your Electron. It allows you to type in and edit any text, and then to arrange it into your own format. Here we present a short program that allows you to do just this.

Using a text editor you will find that it becomes very easy to prepare all kinds of text - letters, memos, reports etc. Your Electron has the potential to be used as a word processing system but at present the software is largely unavailable. We have made a start with the short text editor presented here. Using this you can enter and edit text, saving this to cassette. You could use these new facilities to send text to other Electron users, or to us here at ELBUG, and you could also get a printed version with the new Plus-1 from Acorn or by running the same program on a BBC micro with a printer.

The text editor uses the Electron's own Basic editing facilities to create and edit files of text. These are entered as lines of Basic using the AUTO command, and may be easily saved, loaded and edited (and printed out if you have a Plus-1 expansion and a printer). The program does not allow right-justification or closing up of text, so inserting words could cause a line to overflow on to the next line - in which case the neatest thing to do may be to retype the paragraph.

The function keys are used extensively in this editor, and you may well find it useful to make yourself a simple key strip to put above the keyboard. Type the program in, and save it on your cassette before using it. To use the text editor you will need to run the program first and then use the function keys, pressing Func and key 1 or 2 to start. Either key will initiate the AUTO mode for text entry, but Func and key 2 will put your address as a letter heading (you must replace ELBUG's address with your own here - alter lines 1ØØØ to 1Ø3Ø). Alternatively, pressing Func and key 1 will erase the address from the machine.

Simply type in the text that you require, using 'Return' when you reach the end of a line. If you go over the end of a line, use the Delete key to reduce the line length. As the keys stand, function key 6 will tab along 6 spaces, and function key 8 will tab across to the right side for entering addresses. Function key 9

should be used if you wish to insert a blank line (the Return key will not achieve this). Key 9 leaves the cursor indented 6 spaces to provide a paragraph indentation. Key 3 is not defined, and may be programmed with a word or short sentence that you use often (see the User Guide, page 131, for information on how to program the function keys). The key functions are all listed in the table below.

When you have finished entering text, press the 'Escape' key. Then use the LIST command and editing keys to see your text and make any corrections - just as you would edit a Basic program. Pressing function key 7 will return to mode 6 for better visibility. You can get back into mode 3 by pressing function key Ø, which just executes RUN. To return to entering text after you have escaped from the line numbering mode, you must type AUTO x where x is greater than the number of the last Basic line that holds previously entered text. For this reason it is easier to correct mistakes at the end rather than as you go along.

To save the text created, use the SAVE "name" command, and save and load as a normal program. This means that each text file saved, conveniently has a copy of the editor with it, and may at any time be reloaded and edited further. Remember that when you do this you may need to run the 'program' first to set up the function keys again.

CUSTOMISING FOR YOUR PRINTER

For those of you lucky enough to have an Acorn Plus-1 expansion and a printer, two modes of printout are available. Pressing function key 5 gives a printout with Basic line numbers, for ease of subsequent editing, while key 4 calls a routine which ignores the line numbers. With or without a printer connected, these options will also display the same text on the screen, and shift-control together can be used to control the speed of scrolling. A number of possible options are available with this form of printout. Those chosen will depend on the paper feed mechanism on your printer and the number of lines per page required.

As the program stands it will send a form feed character (check your printer manual to see whether your printer responds to this) after every 31 lines of text. This may be useful if you are printing double spaced text (achieved by typing *FX 6,Ø - assuming that this command has not already been used). If you require a different number of lines printed before a form feed, then alter line 31Ø52 of the program, changing the value 31 to the number required. If you do not want a form feed at all, then erase line 31Ø52 completely.

If you want the program to pause after say 5Ø lines of text have been printed so that you can insert another piece of paper, alter line 31Ø52 to read IF L%>5Ø THEN wait=GET:L%=Ø. Pressing any key when the program pauses in this way, will cause another page to be printed. To increase the spacing between lines, insert 31Ø44 IF ?K%=13 VDU 1Ø,1Ø. Each 1Ø in the VDU command adds an extra line feed after each line of text.

Incidentally, when you select printout without numbers using key 4, you are asked to enter a continuation line number. This is just to allow you to start printout at any line of a page. Thus, as things are presently configured, if you enter 29 here, the program will print out 2 lines of text before doing its first form feed. For the benefit of printers using a continuous roll of paper (and for spacing down the letter head), printout is prefaced with a number of blank lines. These may be removed by deleting lines 950 to 990. At the end of the printout the screen will display a word count (inflated by the effect of double spaces) and the lines in the last page count.

NOTE

Because of the way in which text is stored, it is necessary to avoid Basic reserved words in capitals, and the abbreviations for them. Thus P.O. Box would be printed as PRINTOLDBox on the printout, and as something indecipherable on the numberless printout (since Basic words are stored as tokens).

Table of Key Functions

| Key | Function |
|---|---|
| 1 | Starts the editor without a heading address. |
| 2 | Starts the editor including a heading address. |
| 3 | Printout without line numbers. |
| 4 | Printout with line numbers. |
| 5 | Indentation for start of paragraph. |
| 6 | Returns to mode 6. |
| 7 | Tab for entering address. |
| 8 | Used for inserting a blank line. |
| 9 | Runs the program. |
| Ø | |

PROGRAM NOTE

Because of the complex way that the function keys are set up, it is only possible to put spaces in the definitions by using a space, and not an ASCII value of 32. It is important to type in the exact number of spaces for the key definitions in lines 25160 to 25190. For key 6 there are 6 spaces, and there are also 6 spaces for key 9. Key 8 is entirely up to you, depending on the size of your address but 40 spaces is usually about right.

Yours sincerely

ELBUG

```
   10 REM PROGRAM MINI TEXT EDITOR
   20 REM VERSION E0.2  03/07/84
   30 REM AUTHOR  DAVID GRAHAM
   40 REM ELBUG   AUG/SEPT. 84
   50 REM PROGRAM SUBJECT TO COPYRIGHT
   60 :
  100 GOTO 25000
  950
  960
  970
  980
  990
 1000                          ELBUG,
 1010                          P O Box 5
0,
 1020                          St. Alban
s,
 1030                          Herts.
25000 REM KEY SET ROUTINE
25090 MODE3:*FX18
25100 *KEY0 RUN|M
25110 *KEY1 DELETE1000,1100|M AUTO1000,
10|M
25120 *KEY2 LIST1000,1099|M AUTO1100,10
|M
25140 *KEY4 GOTO31000|M
25150 *KEY5 VDU2|M LIST100,10000|M VDU3
|M
25160 *KEY6 "      "
25170 *KEY7 MODE6|M
25180 *KEY8 "
            "
25190 *KEY9 "|M      "
25300 PROCWINDOW
26999 END
27000 DEF PROCWINDOW
27010 S$="I"+STRING$(9,CHR$32)
27030 PRINTTAB(8,23);:FORS=1TO8:PRINTS$
;:NEXT
27050 VDU28,0,22,79,0
27052 PRINTTAB(2,18);"E L B U G    Mini
 Text Editor"
27054 PRINT"with printout routine by Ia
n Sinclair"
27060 PRINTTAB(0,22);
27070 PRINT"PRESS KEY F1 OR F2 TO START
 (F2 PRINTS ADDRESS)"
27080 PRINT'"OR F4 OR F5 FOR PRINTOUT"
27090 PRINT''''
27200 ENDPROC
31000 REM PRINTOUT WITHOUT NUMBERS
31010 MODE3:PRINT'''
31020 PRINTTAB(2)"Please choose continu
ation or new print- type line number or
 zero.";:INPUT A$:L%=VAL(A$):W%=0
31022 VDU2:N%=(PAGE+1)
31024 REPEAT
31026 N%=N%+?(N%+2)
31028 UNTIL?(N%+1)=10
31029 J%=N%+2
31030 REPEAT
31032 L%=L%+1
31034 FORM%=1TO6
31035 PRINTCHR$(1)" ";
31036 NEXT
31037 FORK%= J%+1 TO N%+?J%-1
31040 PRINTCHR$(?K%);
31042 IF ?K%=32 THEN W%=W%+1
31044 IF ?K%=13 VDU 10
31050 NEXT
31052 IF L%>=31 THEN PRINTCHR$(12):L%=0
31070 N%=N%+?J%:J%=N%+2
31080 UNTIL ?(N%+3)=244:VDU 3
31090 PRINT" Lines of last page- ";L%''
"Word total- "W%
31100 END
```

# HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

REM STATEMENTS IN DATA STATEMENTS - B.Woods

A REM statement may be inserted at the beginning of a data line provided that DATA is followed by a comma and the line is RESTORE'd before use. Here is an example:

```
10 REM staff DATA,NIGE,ALAN,DAVE,****
20 RESTORE 10
30 REPEAT:READ name$:PRINT name$:UNTIL name$="****"
40 END
```

The REM may not otherwise be used on the same line as DATA.

BREAK FROM WITHIN BASIC

It's possible to invoke the same action that takes place when you press the Break key. It can be done by the instruction, CALL !-4, which when written in to a program will give a fast exit from a program and reset the computer all together.

# KAYAK KAPERS

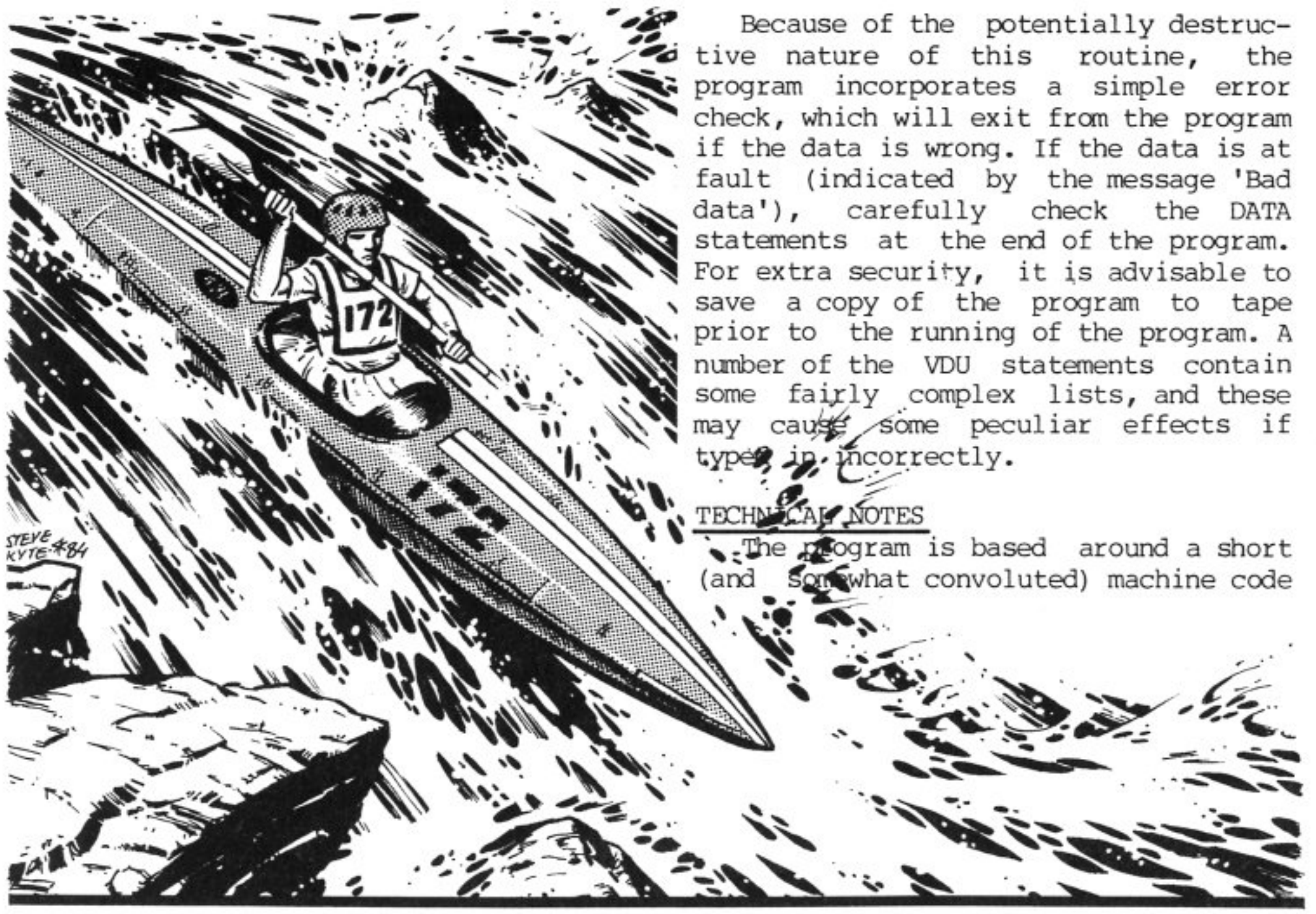
**By 'Surac'**

Kayak Kapers is a superb short game from 'Surac' that features a novel and ingenious sideways scroll to animate the motion of the River El Yako as you struggle to avoid the rocks and survive the everlasting rapids.

Kayak Kapers is a fast action game in which you paddle your flimsy kayak canoe down the fast flowing River El Yako, which is strewn with a large number of rocks and boulders. Your task is to negotiate as far down stream as possible, whilst avoiding the rocks - to collide with them is fatal in such a fragile craft as yours! The river banks are equally hostile and will cause your immediate destruction if you hit them on your way.

The program is delibertately short and demonstrates that good games don't have to involve lengthy programs. Of course, some features of the display have been kept quite simple (the canoe for example) but this is as much to ensure maximum speed as for any other reason.

The game features three levels of play (1 being the easiest, and 3 the hardest), with the current level being selected at the start of each game. Your controls are 'A' to move the canoe up, and 'Z' to move it down the screen. Your current score is displayed at the top, and the latest high score at the bottom. Simple on screen prompts are displayed. The program uses a specially written machine code routine to provide the fast sideways scroll, and this has been shrunk down to the data statements at the end of the program.




Because of the potentially destructive nature of this routine, the program incorporates a simple error check, which will exit from the program if the data is wrong. If the data is at fault (indicated by the message 'Bad data'), carefully check the DATA statements at the end of the program. For extra security, it is advisable to save a copy of the program to tape prior to the running of the program. A number of the VDU statements contain some fairly complex lists, and these may cause some peculiar effects if typed in incorrectly.

TECHNICAL NOTES

The program is based around a short (and somewhat convoluted) machine code

routine which is called to scroll the screen for each cycle of the program. The routine only scrolls the middle section of the screen, and resides at &BØØ onwards - note that this causes the function keys to contain some peculiar definitions. You could easily use this routine in some of your own programs. As presented, the routine does not scroll the whole screen, (the top and bottom 6 lines are left untouched). This could be altered, but this requires a knowledge of screen memory addresses, and it is beyond the scope of this article to cover these values.

In order to incorporate the scroll routine into your own programs, you need lines 27Ø to 34Ø (the procedure to read in the data) and lines 86Ø to 96Ø (the data itself) to be present in your program. Before you can use the scroll routine, you need to call PROCread. The scroll routine is then used by CALL scroll% from within Basic (for example, see the end of line 53Ø).

```
 10 REM PROGRAM ERAPIDS
 20 REM AUTHOR 'SURAC'
 30 REM VERSION E1.Ø
 40 REM ELBUG AUG/SEPT 1984
 50 REM PROGRAM SUBJECT TO COPYRIGHT
 60 :
100 ON ERROR GOTO 980
110 MODE 4:*FX18
120 PROCchars
130 PROCread
140 PROCplay
150 END
160 :
170 DEF PROCchars
180 VDU 23,240,192,240,252,255,255,25
2,240,192
190 VDU23,241,Ø,12,62,110,84,120,16,Ø
200 VDU23,242,Ø,48,120,92,116,88,48,Ø
210 VDU 23,243,Ø,56,92,60,116,88,32,Ø
220 VDU 23,244,Ø,Ø,12,52,44,24,Ø,Ø
230 VDU23,245,Ø,12,62,106,126,44,16,Ø
240 VDU 23,246,Ø,14,58,126,54,28,8,Ø
250 ENDPROC
260 :
270 DEF PROCread
280 RESTORE:scroll%=&BØØ
290 FORI%=ØTO1Ø:READ A$:FORJ%=ØTO7
300 ?(scroll%+J%+I%*8)=EVAL("&"+MID$(
A$,J%*2+1,2))
310 NEXT,:A%=Ø:FORI%=ØTO88
320 A%=A%+scroll%?I%:NEXT
330 IF A%<>&216F THEN PRINT"Bad data"
:END
340 ENDPROC
350 :
360 DEF PROCplay
370 H%=-1:REPEAT
380 VDU 22,5:VDU23,1,Ø;Ø;Ø;Ø;
390 VDU 19,Ø,4,Ø,Ø,Ø
400 VDU 17,129:VDU28,Ø,4,19,Ø,12
410 VDU 28,Ø,31,19,27,12:VDU17,130
420 VDU 28,1,3,18,1,12
430 VDU 28,1,30,18,28,12
440 VDU26,17,128
450 PRINTTAB(4,15)"LEVEL (1-3)";
460 REPEAT T%=GET-48:UNTILT%>ØANDT%<4
470 T%=T%-1:PRINTTAB(4,15)SPC12
480 FORI%=ØTO3:FORJ%=ØTO3:SOUND1,-15,
I%*10+10*J%,2:NEXT,
490 S%=Ø:M%=6:N%=25:X%=15
500 VDU17,1:VDU31,19,6,241:VDU31,19,2
5,241
510 PROCscore(Ø):REPEAT
520 VDU17,128,17,2,31,2,X%,240
530 PROCwalls:CALL scroll%
540 F%=!(&5800+X%*320+32)
550 IF INKEY-66 AND X%<>6 X%=X%-1
560 IF INKEY-98 AND X%<>25 X%=X%+1
570 PROCscore(1):UNTILF%
580 VDU17,128:VDU17,2
590 VDU31,2,X%,240,17,130,17,1,31,4,16
600 PRINT"Press RETURN"
610 *FX15
620 SOUNDØ,-15,7,4
630 REPEAT UNTILGET=13
640 UNTIL FALSE
650 ENDPROC
660 :
670 DEF PROCscore(A%) S%=S%+A%
680 VDU17,131,17,Ø:IF S%>H% H%=S%
690 PRINTTAB(2,2)"Score ";S%
700 PRINTTAB(2,29)"High  ";H%
710 ENDPROC
720 :
730 DEF PROCwalls
740 VDU17,1:P%=RND(3)-2:Q%=RND(3)-2
750 IF T%=1 AND RND(2)=1 P%=Ø:Q%=Ø
760 IF M%+P%<6 P%=Ø
770 IF N%+Q%>25 Q%=Ø
780 IF N%+Q%-P%-M%<8 P%=Ø:Q%=Ø
790 M%=M%+P%:N%=N%+Q%
800 VDU31,19,M%,240+RND(6),31,19,N%,2
40+RND(6)
810 VDU31,19,M%+1,240+RND(6),31,19,N%
-1,240+RND(6)
820 VDU17,3:FORI%=1TOT%
830 IF RND(4-T%)=1 VDU31,19,RND(N%-M%
-1)+M%,240+RND(6)
840 NEXT:ENDPROC
850 :
860 DATA "78A95E8D110BA940"
```

→18

# BUILD A HOUSE

**By Nick Day**

This program will delight all those who enjoy seeing colourful and detailed pictures being generated by computer. We think these results will have particular appeal to younger children, who will watch with growing anticipation to see where the cat will finally appear.

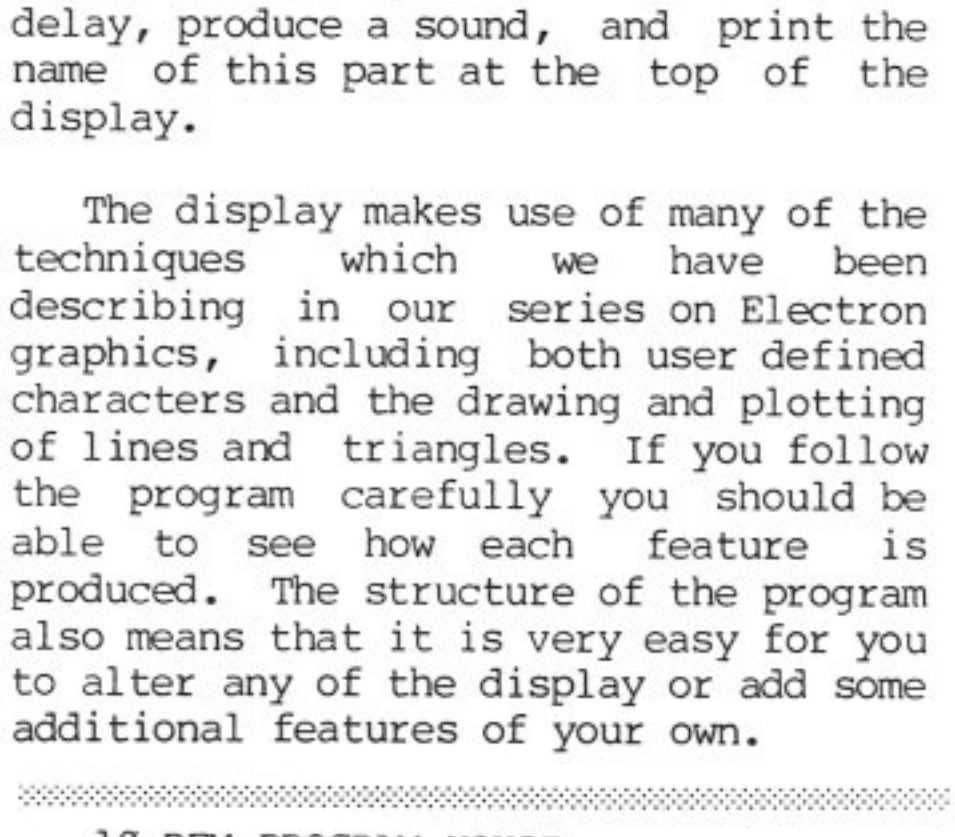
The program listed here runs in mode 2 to provide the maximum number of colours. When you run the program, after saving it first safely to cassette, it will allow you to build up a detailed picture of a house and garden. Each time the space bar (or any other key) is pressed, the next part of the picture will be added to the existing display and the name of this feature will appear at the top of the screen.

The last detail to complete the picture is a white cat which will randomly appear in one of several different places, to provide extra interest.

Once the picture is complete, the whole scene is redrawn as it might appear at night with the moon shining down from above. A few further changes then take place to the picture before the final display is reached, but it would spoil your fun to describe them here.

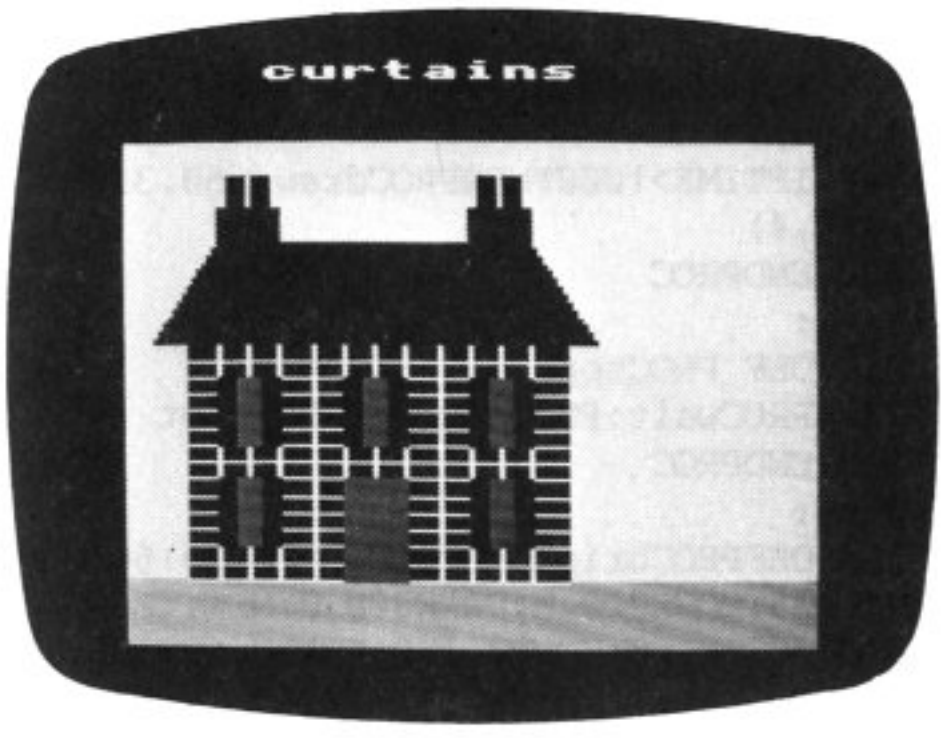
PROGRAM NOTES

If you examine the program listing you will see that its layout is quite obvious, a sequence of instructions naming the next feature and calling a



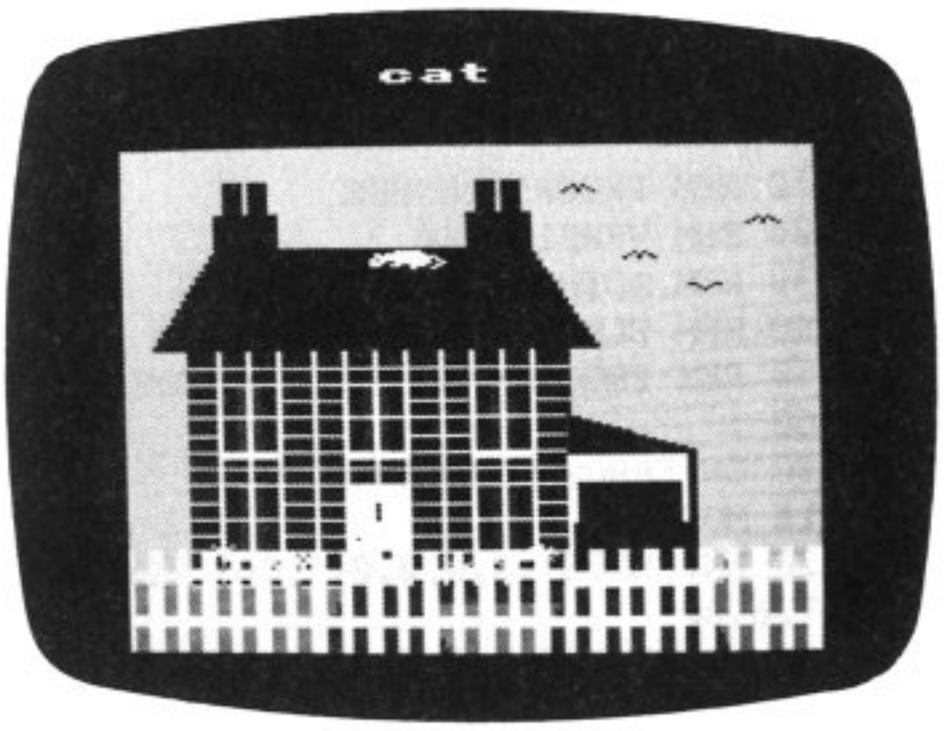

procedure to draw that part of the display. The procedure PROCsopr, which is called every time, in turn calls three further procedures to create a delay, produce a sound, and print the name of this part at the top of the display.

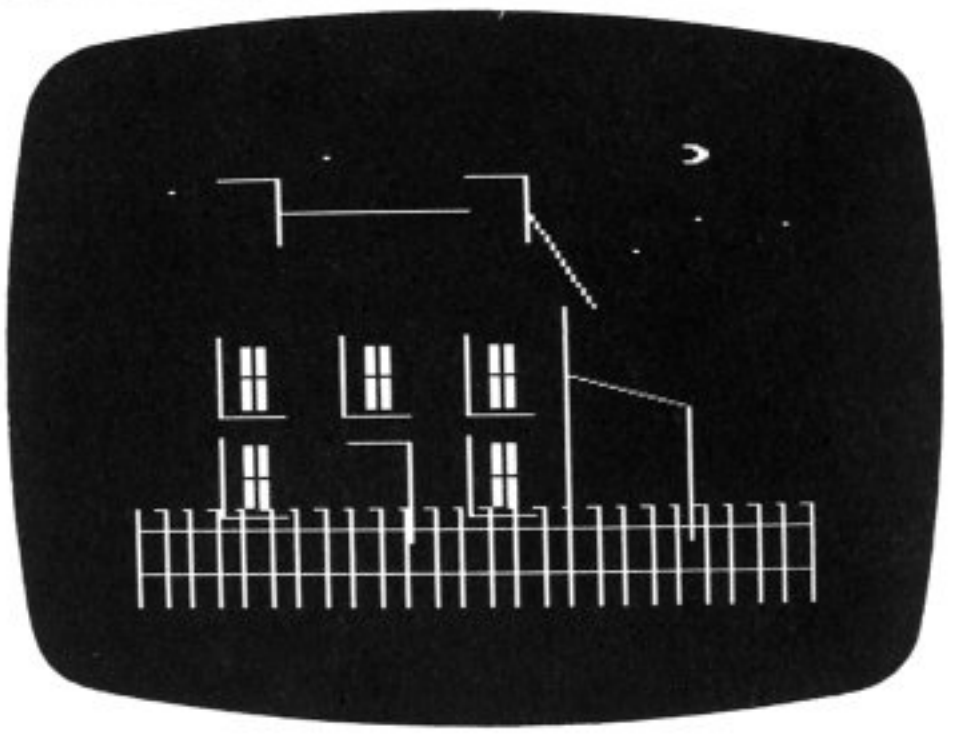
The display makes use of many of the techniques which we have been describing in our series on Electron graphics, including both user defined characters and the drawing and plotting of lines and triangles. If you follow the program carefully you should be able to see how each feature is produced. The structure of the program also means that it is very easy for you to alter any of the display or add some additional features of your own.




```
  10 REM PROGRAM HOUSE
  20 REM VERSION E0.2
  30 REM AUTHOR  N.DAY
  40 REM ELBUG   AUG/SEP. 84
  50 REM PROGRAM SUBJECT TO COPYRIGHT
  60 :
 100 ON ERROR GOTO 1940
 110 MODE2
 120 ENVELOPE1,3,0,0,0,1,1,1,126,-4,0,
-4,126,80
 130 VDU23,1,0;0;0;0;:*FX11,0
 140 *FX4,1
```

```
  150 CLS:VDU24,100;50;1200;800;:VDU28,
0,4,19,3:GCOL0,134:CLG:inc=0
  160 A$="grass":PROCsopr:PROCdraw(100,
50,1200,150,2)
  170 A$="wall":PROCsopr:PROCdraw(200,1
50,800,500,1)
  180 A$="bricks":PROCsopr:PROCbricks
  190 A$="roof":PROCsopr:PROCroof
  200 A$="chimney-stacks":PROCsopr:PROC
stacks(1)
  210 A$="chimney-pots":PROCsopr:PROCpo
ts
  220 A$="doorway":PROCsopr:PROCdoor(5)
  230 A$="windows":PROCsopr:PROCwindows
(5)
  240 A$="curtains":PROCsopr:PROCdrapes
  250 A$="window-frames":PROCsopr:PROCf
rames(7)
  260 A$="window-sills":PROCsopr:PROCsi
lls(250,350):PROCsills(250,200):PROCsil
ls(450,350):PROCsills(650,350):PROCsill
s(650,200)
  270 A$="garage":PROCsopr:PROCgaragero
of(1,0):PROCgardoor(7)
  280 A$="door":PROCsopr:PROCdoor(7)
  290 A$="doorknobs":PROCsopr:PROCknobs
  300 A$="letterbox":PROCsopr:PROCletter
  310 A$="doorknocker":PROCsopr:PROCkno
ck
  320 A$="path":PROCsopr:PROCpaths(5)
  330 A$="drive":PROCsopr:PROCdrive(5)
  340 A$="flowers":PROCsopr
  350 FORf=1TO8:READa,b:PROCflower(a,b)
:NEXT
  360 DATA250,125,325,100,375,110,625,1
00,675,110,750,120,1050,125,1150,100
  370 A$="car":PROCsopr:PROCcar
  380 A$="fence":PROCsopr:PROCfence
  390 A$="birds":PROCsopr:PROCbirds
  400 A$="cat":PROCsopr:PROCcat
  410 now=TIME:REPEATUNTILTIME=now+1000
:CLS
  420 PROCnight:CLS:*FX15,1
  430 TIME=0:REPEAT:PROClightsout:UNTIL
FALSE
```



```
  440 END
  450 :
 1000 DEFPROCdraw(a,b,c,d,e):GCOL0,e:MO
VEa,b:MOVEc,b:PLOT85,a,d:PLOT85,c,d:END
PROC
 1010 :
 1020 DEFPROCdrawN(a,b,c,d,e):GCOL0,e:M
OVEa,d:DRAWc,d:DRAWc,b:ENDPROC
 1030 :
 1040 DEFPROCdrawNu(a,b,c,d,e):GCOL0,e:
MOVEa,d:DRAWa,b:DRAWc,b:ENDPROC
 1050 :
 1060 DEFPROCwait
 1070 *FX15,1
 1080 REPEATUNTILGET=32:ENDPROC
 1090 :
 1100 DEFPROCsound:inc=inc+4:SOUND1,1,(
97+inc),5:SOUND2,1,(113+inc),5:SOUND3,1
,(125+inc),5:ENDPROC
 1110 :
 1120 DEFPROCprint:l%=LEN(A$)/2:PRINTTA
B(9-l%)A$:FORW=1TO500:NEXT:ENDPROC
 1130 :
 1140 DEFPROClightsout
 1150 IFTIME>200THENPROCdraw(250,200,35
0,300,4)
 1160 IFTIME>400THENPROCdraw(650,200,75
0,300,4)
 1170 IFTIME>600THENPROCdraw(250,350,35
0,450,4)
 1180 IFTIME>800THENPROCdraw(450,350,55
0,450,4)
 1190 IFTIME>1000THENPROCdraw(650,350,7
50,450,4)
 1200 ENDPROC
 1210 :
 1220 DEF PROCsopr
 1230 PROCwait:PROCsound:PROCprint
 1240 ENDPROC
 1250 :
 1260 DEFPROCbricks:GCOL0,3:FORB=150TO4
75STEP25:MOVE200,B:DRAW800,B:NEXT:FORB=
250TO750STEP50:MOVEB,150:DRAWB,500:NEXT
:ENDPROC
```

```
 1270 :
 1280 DEFPROCroof:PROCdraw(250,500,750,
650,0):PROCgables:ENDPROC
 1290 :
 1300 DEFPROCgables:MOVE750,500:MOVE750
,650:PLOT85,850,500:MOVE250,650:MOVE250
,500:PLOT85,150,500:ENDPROC
 1310 :
 1320 DEFPROCstacks(c):PROCdraw(250,600
,350,700,c):PROCdraw(650,600,750,700,c)
:ENDPROC
 1330 :
 1340 DEFPROCpots:PROCdraw(270,700,290,
750,0):PROCdraw(310,700,330,750,0):PROC
draw(670,700,690,750,0):PROCdraw(710,70
0,730,750,0):ENDPROC
 1350 :
 1360 DEFPROCwindows(c):PROCdraw(250,20
0,350,300,c):PROCdraw(250,350,350,450,c
):PROCdraw(450,350,550,450,c):PROCdraw(
650,200,750,300,c):PROCdraw(650,350,750
,450,c):ENDPROC
 1370 :
 1380 DEFPROCdrapes:PROCcurtains(250,45
0):PROCcurtains(450,450):PROCcurtains(6
50,450):PROCcurtains(250,300):PROCcurta
ins(650,300):ENDPROC
 1390 :
 1400 DEFPROCcurtains(x,y):PROCdraw(x,y
-100,x+25,y,4):PROCdraw(x+70,y-100,x+10
0,y,4):ENDPROC
 1410 :
 1420 DEFPROCframes(c):GCOL0,c:MOVE250,
450:DRAW350,450:DRAW350,350:DRAW250,350
:DRAW250,450:MOVE300,450:DRAW300,350:MO
VE250,400:DRAW350,400
 1430 MOVE450,450:DRAW550,450:DRAW550,3
50:DRAW450,350:DRAW450,450:MOVE450,400:
DRAW550,400:MOVE500,350:DRAW500,450
 1440 MOVE650,450:DRAW750,450:DRAW750,3
50:DRAW650,350:DRAW650,450:MOVE700,350:
DRAW700,450:MOVE650,400:DRAW750,400
 1450 MOVE650,300:DRAW750,300:DRAW750,2
00:DRAW650,200:DRAW650,300:MOVE650,250:
DRAW750,250:MOVE700,200:DRAW700,300
 1460 MOVE250,300:DRAW350,300:DRAW350,2
00:DRAW250,200:DRAW250,300:MOVE250,250:
DRAW350,250:MOVE300,200:DRAW300,300:END
PROC
 1470 :
 1480 DEFPROCsills(x,y):PROCdraw(x,y-10
,x+100,y,7):ENDPROC
 1490 :
 1500 DEFPROCgarageroof(c,r):PROCdraw(8
00,340,1000,350,c):PROCdraw(990,150,100
0,350,c):MOVE800,400:MOVE800,350:GCOL0,
r:PLOT85,1000,350:ENDPROC
 1510 :
 1520 DEFPROCgardoor(c):PROCdraw(800,15
0,990,340,c):ENDPROC
 1530 :
 1540 DEFPROCdoor(c):PROCdraw(450,150,5
50,300,c):ENDPROC
 1550 :
 1560 DEFPROCknobs:GCOL0,0:PLOT69,470,2
20:PLOT69,900,170:ENDPROC
 1570 :
 1580 DEFPROCletter:PROCdraw(485,195,51
5,200,0):ENDPROC
 1590 :
 1600 DEFPROCknock:PROCdraw(497,250,503
,275,0):ENDPROC
 1610 :
 1620 DEFPROCpaths(c):PROCdraw(450,50,5
50,150,c):PROCdraw(550,125,800,150,c):M
OVE450,150:MOVE450,50:PLOT85,400,50:MOV
E550,150:MOVE550,50:PLOT85,600,50:ENDPR
OC
 1630 :
 1640 DEFPROCdrive(c):PROCdraw(800,50,1
000,150,c):MOVE800,150:MOVE800,50:PLOT8
5,750,50:MOVE1000,150:MOVE1000,50:PLOT8
5,1050,50:ENDPROC
 1650 :
 1660 DEFPROCflower(x,y):MOVEx,y:GCOL0,
2:DRAWx,y+70:VDU23,245,0,0,0,0,16,0,0,0
:VDU23,246,16,84,56,238,56,84,16,0:VDU5
:GCOL0,0:MOVEx-20,y+90:VDU245:VDU8:GCOL
0,3:VDU246:GCOL0,0:MOVE500,600:VDU4:END
PROC
 1670 :
 1680 DEFPROCcar:PROCdraw(825,100,850,1
50,0):PROCdraw(950,100,975,150,0):PROCd
raw(800,150,1000,160,7):PROCdraw(810,16
0,990,240,4):PROCdraw(820,240,980,300,4
):PROCdraw(830,250,970,290,0)
 1690 PROCdraw(816,160,824,200,1):PROCd
raw(980,160,984,200,1):ENDPROC
 1700 :
 1710 DEFPROCfence:GCOL0,7:FORX=88TO120
0STEP44:PROCdraw(X,50,X+15,200,7):NEXT:
PROCdraw(100,160,1200,170,7):PROCdraw(1
00,90,1200,100,7):ENDPROC
 1720 :
 1730 DEFPROCbirds:VDU4:VDU5:VDU23,240,
0,0,40,84,146,0,0,0:VDU23,241,0,0,198,4
0,16,0,0,0:GCOL0,0:MOVE800,750:VDU240:M
OVE900,650:VDU240:MOVE1000,600:VDU241:M
OVE1100,700:VDU240:MOVE600,600:VDU4:END
PROC
 1740 :
 1750 DEFPROCcat:VDU4:VDU5:VDU23,242,0,
7,95,127,191,231,204,0:VDU23,243,0,192,
224,248,244,242,228,136:GCOL0,7:PROCpos
:MOVEc,t:VDU242:VDU243:GCOL0,0:MOVE500,
600:ENDPROC
 1760 :
 1770 DEFPROCpos:p=RND(4):c=500:t=600:I
Fp=1THENc=810:t=375
 1780 IFp=2THENc=250:t=725
```

```
 1790 IFp=3THENc=500:t=650
 1800 ENDPROC
 1810 :
 1820 DEFPROCnight:GCOL0,132:CLG
 1830 PROCroof:PROCstacks(4):PROCpots:P
ROCdrawN(450,150,550,300,6):PROCwindows
(3):PROCdrapes:PROCgarageroof(0,0):PROC
frames(4)
 1840 GCOL0,6:MOVE250,700:DRAW350,700:D
RAW350,600:MOVE350,650:DRAW650,650:MOVE
650,700:DRAW750,700:DRAW750,600:MOVE750
,650:DRAW850,500:MOVE800,500:DRAW800,15
0:MOVE800,400:DRAW1005,350:DRAW1005,150
 1850 PROCdrawNu(240,190,350,310,6):PRO
CdrawNu(240,340,350,460,6):PROCdrawNu(6
40,190,750,310,6):PROCdrawNu(440,340,55
0,460,6):PROCdrawNu(640,340,750,460,6)
 1860 FORX=88TO1200STEP44:PROCdrawN(X,5
0,X+15,200,6):NEXT:PROCdrawN(100,160,12
00,170,6):PROCdrawN(100,90,1200,100,6)
 1870 VDU5:VDU23,248,0,112,24,12,12,12,
24,112:VDU23,249,0,0,0,0,16,0,0,0:GCOL0
,7:MOVE1000,750:VDU248:MOVE150,700:VDU2
49:MOVE400,750:VDU249:MOVE900,600:VDU24
9:MOVE1000,650:VDU249:MOVE1150,640:VDU2
49
 1880 VDU4:COLOUR0:ENDPROC
 1890 :
 1900 DEFPROClo:SPL$="":FORch=1TOLEN(SP
$):IFMID$(SP$,ch,1)="-"SPL$=SPL$+"-":GO
TO1920
 1910 SPL$=SPL$+CHR$(ASC(MID$(SP$,ch,1)
)+32)
 1920 NEXT:ENDPROC
 1930 :
 1940 ON ERROR OFF
 1950 *FX4
 1960 *FX12
 1970 MODE 6:IF ERR=17 END
 1980 REPORT:PRINT" at line ";ERL
 1990 END
```

14 ←

```
 870 DATA "8D100BA900A00F99"
 880 DATA "0000B810FA18AD10"
 890 DATA "0B69408D100BAD11"
 900 DATA "0B69018D110BC979"
 910 DATA "D0E1A279A95F8D42"
 920 DATA "0B8D450BA9808D44"
 930 DATA "0BA9908D410BA000"
 940 DATA "B90000990000C8D0"
 950 DATA "F7EE420BEE450BEC"
 960 DATA "450BD0EA58604446"
 970 :
 980 ON ERROR OFF
 990 MODE 6
1000 IF ERR<>17 REPORT:PRINT" at line
";ERL
1010 END
```

# POSTBAG POSTBAG POSTBAG POSTBAG POSTBAG POSTBAG

SQUARE WAVES

Dear Sir,

After reading the article in the March issue of ELBUG about using BBC micro programs on the Electron, I experimented on my machine to see if the ADVAL statement had any effect and discovered some interesting results. I wrote this program to demonstrate these results graphically:

```
10  MODE 0
20  FOR X=0 TO 1279
30  DRAW X,ADVAL(x)/100
40  NEXT X
```

This produced a very rough square wave pattern [very rough - Ed.]. Although I cannot explain these results, they seem to disprove the statement on page 28 (table 1) of the article.

Tim Stephenson

Reply:

We are not sure either of the reasons for these results. However, it should be noted that even on a BBC micro ADVAL is only valid with values from 0 to 4, and using ADVAL in the way described produces similar effects on that machine.

SCREEN FREEZER

Dear Sir,

Your "Screen Freezer" program (ELBUG issue 7) is excellent and I thought you would like to know that "Felix in the Factory" from Program Power freezes very well indeed.

I enjoy your magazine tremendously (only wish it were longer).

Tom Ede

Reply:

If other readers write to us we might publish a table of all the programs that you tell us can be 'frozen' in this way. Regretfully perhaps, we have no plans at present to increase the number of pages in the magazine.

# MONITORS FOR THE ELECTRON

**Reviewed by Steve Fox**
**and Hugh Brown-Smith**

In a previous article Steve Fox discussed some of the points to think about when choosing a TV or monitor to use with your Electron. This month Steve turns his attention to two of the cheaper colour monitors you might be tempted to consider buying, and gives you his expert opinion. Hugh Brown-Smith describes the rather more expensive Microvitec Cub.

Grundig 'Super Colour' C2402 GB TV/Monitor supplied by Newark Video Centre, 108 London Road, Balderton, Newark Notts. NG24 3AQ. Price (to ELBUG members) £260 inc. VAT and carriage.

Reviewer : Steve Fox.

GRUNDIG "SUPER COLOUR" TV/MONITOR

This 14" Portable TV Receiver is of German origin and is distributed in the U.K. by Grundig International of Rugby. It is very modern in design and incorporates infra-red remote control of channel selection, volume, brightness, colour, loud-speaker muting and reset. The receiver is able to store, while switched off, tuning data for up to 30 channels, the same tuning buttons servicing all channels. A miniature jack is supplied to allow TV sound to be fed to an earpiece (or tape recorder, though the quality of recording is likely to be poor due to overloading) and a twin-rod V-aerial is fitted to the case (it may be unplugged if an external aerial is used). The case of the receiver is black plastic incorporating a recess at the top for lifting: its weight is comparatively modest (about 25lb). A flap at the front gives access to the normal presets, including contrast. When the flap is closed it depresses a button which brings into action an automatic frequency correction device which prevents the receiver drifting off tune.

A straightforward modification of the receiver is carried out (with the approval of Grundig) by certain dealers to enable direct RGB connection, via DIN plugs to the BBC micro or Electron. The receiver being reviewed was modified by Newark Video Centre, 108 London Road, Balderton, Newark, Notts, NG24 3AQ. The changeover from TV to RGB and back is controlled by a switch on the back.

First impressions of the monitor, viewing a TV picture, were very favourable. Definition was crisp, geometry good and the colours natural. I turned the colour control to the "off" end of the range and noted that the resulting black and white picture was neutral in tint. There wasn't any suggestion of coloured edges and the black level remained constant whether the scene was light or dark. I then tried the RGB mode. Computer-generated screens tend to be over-colourful and this effect can be quite objectionable on certain picture-tubes whose screen phosphors are of strident hue. The tube used by Grundig is excellent in this respect; the primary colours are strong but pure.

The test programs described in the previous article confirmed these impressions. The white grille and dots pattern showed no coloured fringes and no perceptible geometrical distortions. The screen margins on this receiver are practically rectangular, only the extreme corners being rounded. The scanning was also correctly adjusted (whether by the factory or by Newark Video Centre) so that the entire picture was visible. With TV receivers, this job all too often has to be carried out by the purchaser, as the margins are often rounded and the factory usually sets the picture so that parts of it are lost at the top, bottom and edges.

The 'colour bars' test program showed a generally smooth transmission from one colour to another without any serious overshoot or undershoot, although the Grundig is not as good as the Microvitec Cub in this respect.

The 'small print' test showed that the Grundig is as tolerable for word processing as any other standard resolution colour monitor. Its excellent convergence (absence of coloured edges all over the screen area) helps a lot and provided the brightness and contrast are carefully adjusted I would think it usable for periods of up to, say, an hour without too much eye-strain.

The EHT Regulation test is a severe one, and all monitors except very expensive professional ones must be expected to show geometric changes in the picture as the CRT beam current rises or falls. When the raster is white the current drain is at a maximum and the EHT is reduced. The Grundig shows a small increase in the raster size under these conditions, with a bowing inwards of the edges (pincushion distortion), but the effect is not excessive nor abnormal.

No interaction of the RF receiving circuitry was perceptible when in the RGB mode, but if TV is selected without switching off or disconnecting the micro, synchronizing problems appear.

A minor but annoying quirk of the Grundig circuit is that if the monitor is switched on at the mains plug it goes into "standby" condition and does not display a picture (even in RGB) until a channel is selected with the remote controller. This does not happen if the receiver is switched in with its own on/off switch.

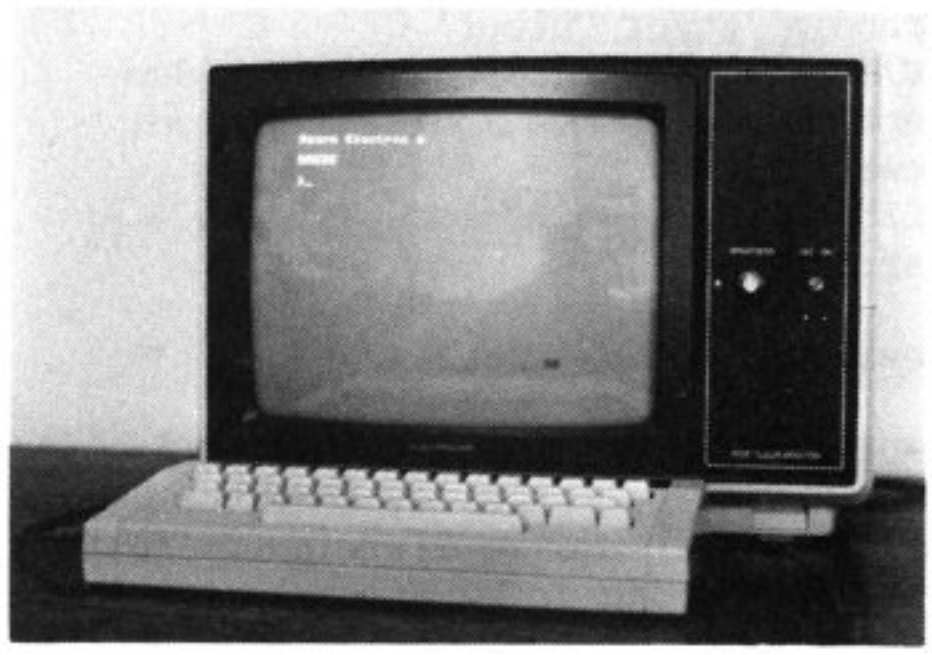
The TV receiver as such is sensitive and gives a good clean picture in good reception areas, although results using the 'V' aerial provided will of course be unpredictable. My only criticism is that the AFC circuit mentioned in my first paragraph introduces a small but perceptible amount of noise (or "grain") into the TV picture and to avoid this the control flap has to be left open.

Newark Video Centre sell this monitor (carriage paid) to ELBUG members at £260 - a discount of £15 on the normal price of £275. Other firms appear to charge more.

It is an excellent monitor in its class and I can recommend it as very good value.

Electrohome ECM1302-1 RGB Monitor
supplied by Opus Supplies Ltd.,
158, Camberwell Road, London SE5 0EE.
Price £221.89 inc. VAT and carriage.

Reviewer : Steve Fox.



ELECTROHOME ECM 1302 RGB MONITOR

These monitors are manufactured for Electrohome Electronics, a Canadian firm, by the Japanese JVC company. They are imported into the UK by Opus Supplies Limited who sell direct to the public. They are available in two forms: "medium" and "high" resolution. The main difference between the two is that the medium-resolution model uses a standard picture tube of the type used in domestic TVs whereas the high resolution version uses a screen of finer structure - although still not as fine as that used in more expensive monitors.

The monitor is fitted with two inputs, one a subminiature "D" connector for the Apple II, the other a 7-pin DIN connector. Connections on the latter require positive sync pulses. The BBC/Electron normally feeds negative syncs although a link can be changed over to reverse the polarity. However, Opus Supplies save you the trouble by supplying a hybrid lead which feeds negative syncs into the Apple input and the other signals into the DIN connector. This is an effective if slightly untidy solution to the problem.

The monitor case is two-tone plastic, pleasant and robust, and the only accessible controls are brightness and on/off. First impressions when I ran an arcade game on the "medium" (i.e. standard) monitor were of a good bright picture very suitable to the subject.

However, when I ran the monitor tests, it became evident that all was not well. The white grille and dots showed coloured fringes at the edges of the picture and the vertical centering was incorrect. Opus Supplies had kindly furnished me with service data, including a circuit diagram, so I removed the back of the set and found that the centering could be corrected (near enough) by changing a 3-position link. But I was less successful in my efforts to reconverge the monitor and could not entirely eliminate the coloured fringes. The next problem was that the whites were pink. I corrected this to my satisfaction by adjusting the red and green channel-gain presets.

When I ran the 'colour bar' test program signal I noticed a black band to the left of the green bar and a bright band to the right of it and other bars showed a similar but less marked effect. I looked at the circuit diagram but could find no trimming adjustments to correct this.

The "fine print" test was of course marred by the colour fringes, although not so severely as to make the letter illegible. The monitor showed no excessive geometric changes when I ran the EHT regulation test.

Viewing pictures on the screen, definition was subjectively good, being at times emphasized by the bright green lines. However, fine green lettering was quite difficult to read.

Further enquiries have convinced me that the monitor reviewed was not exceptional and that its shortcomings are due to a pretty basic design rather than to sub-standard components. That said, as I have already indicated, the monitor's performance was capable of improvement by a qualified engineer, as some of the presets were not properly adjusted.

At the inclusive price of £221.89, this monitor represents fair value for money, though the well known and impressive Microvitec costs only a little more.

I have not checked the "high resolution" version, but as its design very closely resembles the other, I have no doubt that it suffers from the same problems.

**Microvitec Cub 1431 MS Colour Monitor**
**Microvitec PLC., Futures Way,**
**Bolling Road, Bradford,**
**West Yorkshire, BD4 7TU.**
**Price £238.85 inc. VAT and carriage.**

**Reviewer : Hugh Brown-Smith**

## MICROVITEC CUB

The Microvitec Cub has become something of a standard for colour monitors used with the BBC. This is largely due to the volume of units supplied to educational establishments under the D.O.I. scheme. The fact that so many have been sold has perhaps biased many home users into choosing the Cub. So is this the best available, or should one consider other alternatives?

The monitor itself looks very smart and robust, the case being made entirely of sheet metal finished in a two tone brown. The lack of a carrying handle could make it awkward to move the monitor and computer together, but this does not present a great problem. Strength and robustness seem to have been a theme of the design, giving the appearance of being able to withstand easily all but the most determined knock. The 14 inch screen occupies almost the whole of the front panel with a small indicator to the bottom right showing when the monitor is on.

The only controls available outside the case are an on/off switch and contrast, both situated on the rear panel, but within easy reach if you want to make adjustments. The lack of a brightness control does not present any problems as the contrast will allow more than adequate adjustment over the appearance of the screen. The factory set condition of the internal controls seems to be accurate and the picture

position is good, although on the review model the picture was very slightly off centre. Also on the rear panel is the RGB input via a six pin Din plug. A lead is supplied with the monitor for use with BBC or Electron machines.

Linearity of these monitors is excellent as is the colour definition. 80 column definition is far better than a TV but lacking slightly in readability. This is due to slight fringing but without a 'High Resolution' tube such a monitor cannot be expected to produce an image suitable for long use without causing potential eyestrain.

The main problem with the Microvitec is an annoying flicker which can be quite off-putting if the screen is viewed at close distances for prolonged periods. This seems inherent on all Cubs although some are better than others. If you decide to buy one, try to go to a dealer and see the particular one you are going to buy working first. The flicker is most apparent at the top of the screen although it is also present in the middle.

The test programs published in ELBUG issue 7 were used to put the monitor through its paces, and the results are described below.

The white bar and dot test revealed that the top corners of the screen were adjusted badly and set just out of view. This is not the case on all models so again it is worth seeing one first.

The colours shown in the colour bar test were all of good definition and clarity. There was also no problem where two colours met.

The 80 column text test suffered from slight fringing, as described above, but no more than would be expected and nothing to worry about.

No problems were experienced with the EHT Regulation test. In fact the performance here was better than expected with only a very marginal distortion of geometry visible just in the centre of the outer edges.

All in all the Microvitec represents very good value for money combined with a good reliable service. It seems quite rare for problems to occur, and because of the quantity in service it is not difficult to find someone to undertake repairs.

[We have included on this month's magazine cassette a further test program called TVTEST, which displays a colour test card on the screen. This combines several of the features of the previous test programs, but it is a program which would be tedious to type in accurately from the keyboard. For this reason we are not printing the program in the magazine. - Ed.]

MONOCHROME MONITORS

In this review, all the monitors have been of the colour type. Monochrome monitors are also available which have the advantage of being of very high resolution and relatively cheap. Such monitors (for example those by Novex) are available in a number of different screen types from the black and white through green and black to amber and black. The purpose of these alternative colours is to relieve eye strain and the final choice would obviously be based on personal preference.

If you are particularly interested in a monitor capable of displaying 80 columns of text on the screen (for word processing for example) then a monochrome monitor may be a better choice and the different colours still appear as different shades of grey. You could still use your domestic TV if you wanted a colour picture for games. Prices range depending on make and resolution, but as a guide you could expect to pay £90 - £110 for a high resolution model.

## HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

MODE SELECTION ON BREAK

*FX255,n selects the screen mode when you press the Break key, where n is the mode that you want with 8 added to it.

# ELECTRON GRAPHICS (Part 9)

by Mike Williams

In last month's article in this series we looked at some of the ways of manipulating the colours used in graphics displays on the Electron using the GCOL and VDU19 instructions. This month we shall concentrate solely on further uses of GCOL, and show how you can use this to produce a wide variety of shades in the different colours.

First of all let's just look at the versions of GCOL as documented in the User Guide. The format of the GCOL instruction is:

GCOL p,c

where 'p' is a number indicating the plotting mode to be used, and 'c' is a number which specifies a colour (with 128 added for a background colour). Last month we concentrated on the case where p=3, known as Exclusive-OR plotting. This is particularly valuable when we want to produce a moving graphics display. We will now examine briefly each of the different plotting modes in turn. In the examples that follow the colours referred to will be the default colours (black, red, yellow, white) for a four colour mode (modes 1 and 5). You should be able to work out yourself what the colours would be in the other graphics modes (Ø, 2, 4).

p=Ø: In this mode we simply plot the colour specified by GCOL regardless of any colours already on the screen. Thus GCOLØ,1 will plot in red and GCOLØ,2 will always plot in yellow.

p=1: In this plotting mode it is the logical OR of the new colour and the previous screen colour which will be displayed. Type in the following program:

```
10 MODE 5: GCOL Ø,1
20 PLOT 5,300,300
30 GCOL 1,2: DUMMY=INKEY(100)
40 PLOT 5,Ø,Ø
50 END
```

First of all you will see a red line (i.e. colour 1 as specified by the GCOL in line 10), and then one second later you will get a white line, though one might at first sight expect a yellow one (colour 2). Initially each point on the line was at logical colour 1 (red).

We then set the logical colour to 2 (yellow), but with OR plotting, '1 OR 2' has a logical value of 3 (you can verify this by typing

PRINT 1 OR 2 <return>

in Basic). This is why we get a white line (logical colour 3) instead of a yellow one.

p=2: This plotting mode uses the logical AND of the two colours. If we change line 30 above to read:

```
30 GCOL2,2: DUMMY=INKEY(100)
```

we can see the effect of this. Colour 1 (red) AND colour 2 (yellow) gives colour Ø (black) because '1 AND 2' has the logic value zero. You can again verify this as before (PRINT 1 AND 2). Thus after one second is up the line disappears (changed to black, the same as the background) even though our specified foreground colour is yellow.

p=3: This is Exclusive-OR plotting (abbreviated EOR) as described last month. You should be able to work out what happens if you change line 30 to include GCOL3,2 (see end of article for the answer if you are not sure).

p=4: This ignores the current colour setting completely and merely inverts the colour already present. Thus if you change line 30 to use plotting mode 4 the red line will change to yellow because the inverse (logical NOT) of colour 1 (red) is colour 2 (yellow). In binary, inverting a number changes 1 to 0 and 0 to 1 throughout.

GCOLMANIA

The five plotting modes described above, with values from 0 to 4, are the only ones described in the User Guide. Technically this number is stored as a single byte of memory and is thus potentially capable of taking any value from 0 to 255. If you try this out you will find that it is possible to produce all sorts of exciting striped colours some of which when viewed from a distance appear as shades of the standard 16 colours (includes flashing colours of course). The following program allows you to explore this new world quite easily, and to find some new effects that you can incorporate into your own programs.

When you run the program you will be asked which graphics mode you want to use (0, 1, 2, 4, 5). The screen will then be divided up into rectangular areas depending on the number of colours available in each mode. Thus the two colour modes will display two rectangles, the four colour modes will show four, and mode 2 will display 16 separate areas. Each rectangle represents one of the colours for that mode numbered from left to right and top to bottom. Thus in a four colour mode the top two squares are colours 0 and 1, and the bottom two squares are colours 2 and 3.

As listed the program sets the plotting mode to 0 and displays all the colours possible in the selected graphics mode on the screen. Pressing the space bar (or any other key) will increase the plotting mode by 8 and re-display the colours in this new mode. Continuing to press the space bar will cycle through many of the different effects possible until the value of the plotting mode exceeds 255.

Each screen display will also show the current plotting mode for further reference. Stepping through the plotting modes in steps of 8 should mean that all the effects are functionally equivalent to plotting mode 0, thus avoiding any further complications through the use of the logical operations already described. If you want to explore intermediate values then just change the step size in the program at line 1150.

Next month's article will be the last article in this series, and will show you how to use GCOL and the VDU19 instructions again, this time to produce some fast moving animation.

[Exclusive-OR plotting gives colour 3 (white) in the earlier example.]

PROGRAM NOTES

These notes are included for those who are interested in the details of the program. The main program, from line 10 to 280, simply displays some information on the screen, and determines the graphics mode to be used before calling the procedure PROCdisplay2 at line 1130 to produce the entire display. The main program also calls the procedure PROCinit, which sets up and initialises four arrays to specify the number of colours, and the number of display rows and columns for each mode. By storing this information it is very easy for the program to use the same routines subsequently to create the display, regardless of which graphics mode is chosen.

The procedure PROCdisplay2 is simply a loop which cycles through each GCOL plotting mode and calls a further procedure PROCdisplay to create the display itself.

The procedure PROCdisplay in turn creates each rectangular area as a window which is then filled with a background colour using CLG (see part 5 in this series for a more detailed description of this technique).

```
10 REM Program GCOLMANIA
20 REM Version E1.1
30 REM Author Mike Williams
40 REM ELBUG Aug/Sept 1984
50 REM Program subject to copyright
```

```
   60 :
  100 MODE 1
  110 ON ERROR MODE 6:PROCerror:END
  120 PROCinit
  130 VDU19,2,6,0,0,0
  140 REPEAT:CLS
  150 COLOUR1:PRINTTAB(16,2)"GCOLMANIA"
  160 COLOUR2:PRINTTAB(19,6)"by"
  170 PRINTTAB(14,9)"Mike Williams"
  180 COLOUR3
  190 INPUTTAB(1,16)"Which mode to disp
lay";mode%
  200 UNTIL mode%<6 AND INSTR("01245",S
TR$(mode%))
  210 MODE mode%:M%=-1
  220 VDU23,1,0;0;0;0;
  230 REPEAT:M%=M%+1:UNTIL mode%(M%)=mo
de%
  240 x%=column%(M%):y%=row%(M%)
  250 PROCdisplay2(x%,y%)
  260 MODE 6
  270 END
  280 :
 1000 DEF PROCinit
 1010 LOCAL M%
 1020 DIM mode%(4),colours%(4),column%(
4),row%(4)
 1030 FOR M%=0 TO 4
 1040 READ mode%(M%),colours%(M%),colum
n%(M%),row%(M%)
 1050 NEXT M%
 1060 DATA 0,2,2,1
 1070 DATA 1,4,2,2
 1080 DATA 2,16,4,4
 1090 DATA 4,2,2,1
 1100 DATA 5,4,2,2
 1110 ENDPROC
 1120 :
 1130 DEF PROCdisplay2(x%,y%)
 1140 LOCAL gcol%,z%
 1150 FOR gcol%=0 TO 255 STEP 8
 1160 PROCdisplay(gcol%,x%,y%)
 1170 z%=GET
 1180 NEXT gcol%
 1190 ENDPROC
 1200 :
 1210 DEF PROCdisplay(g%,x%,y%)
 1220 LOCAL colour%,incx%,incy%,I%,J%
 1230 incx%=1280/x%:incy%=1024/y%
 1240 FOR J%=y%-1 TO 0 STEP -1
 1250 FOR I%=0 TO x%-1
 1260 VDU24,I%*incx%;J%*incy%;(I%+1)*in
cx%-1;(J%+1)*incy%-1;
 1270 colour%=y%*(y%-1-J%)+I%
 1280 GCOL g%,colour%+128:CLG
 1290 NEXT I%,J%
 1300 PRINTTAB(6,2)"GCOL ";gcol%;
 1310 ENDPROC
 1320 :
 1330 DEF PROCerror
 1340 ON ERROR OFF
 1350 IF ERR<>17 THEN REPORT:PRINT" at
line ";ERL
 1360 ENDPROC
```

# HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

TOP LINE OF TEXT - D. Robinson

If you have difficulty reading the top line of text on your television then you can define a text window to stop the Electron using the top line. This command is slightly different for each mode to compensate for the different screen heights e.g.

```
MODE 6:VDU 28,0,24,39,1
MODE 5:VDU 28,0,31,19,1
```

SHADED LETTERING FOR TITLES - W.R.Powers

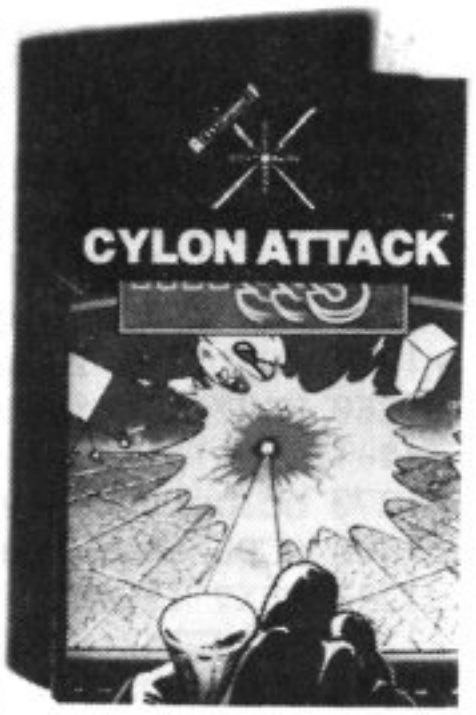
To make your titling in programs more attractive try the following procedure which starts at line 100 (the rest is just to demonstrate it).

```
  10 MODE 1:COLOUR 2
  20 PROCshade("Elbug for the Electron",260,800)
  30:
 100 DEF PROCshade(A$,X,Y)
 110 VDU5
 120 VDU19,0,5,0,0;:VDU19,1,0;0;
 130 GCOL1,3:MOVEX,Y:PRINT A$
 140 GCOL1,1:MOVEX-8,Y-8:PRINT A$
 150 ENDPROC
```

This uses VDU5 to print text at the graphics cursor, printing the letters a second time offset by 8 graphics units.

# More Games Reviewed

Name : Cylon Attack
Supplier : A & F Software
Price : £7.90
Reviewer : David Fell
Rating : ****

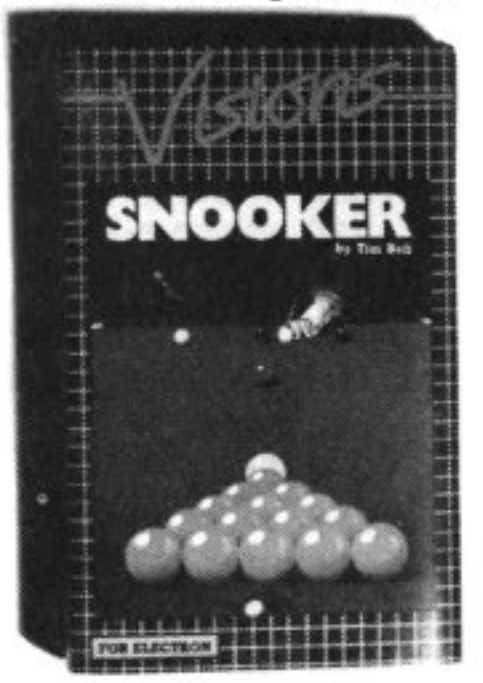



This game involves you leaving the sanctity of your mothership to ward off a variety of alien space ships, all of whom are intent upon destroying your space craft. The screen provides you with a view from the cockpit of your space ship, and you are provided with an indication of your fuel, shield and laser status. At the top of the screen is a similar display for your mothership, showing how many fuel refills there are left, and how many more hits can be taken on the mothership (hits are only taken on the mothership when you dock with it, and refuel).

The game features some attractively designed shapes, and presents a well 'graded' game. The first round is quite easy, and a little patience will allow you to finish this round quite easily. The later rounds, however, are amazingly difficult, and require some very subtle game play to master. The high score table provided allows for 50 scores, and can be saved and loaded from tape when necessary. There is an option to select from either two key layouts, or joysticks, and details of a high score competition are given with the packaging.

Title: SNOOKER
Supplier: Visions
Price: £7.95
Reviewer: Mike Williams
Rating: *

I would not claim much expertise in the practice of snooker, but even to my untrained eye, the quality of play available in this version of the game is poor. The balls move in a most unrealistic way, often appearing to stop dead as though a bunker has suddenly materialised on the green of the snooker table, while some of the angles and rebounds would drive a professional demented.




Colourful this game may be, but realistic it definitely is not. This is a pity because the game of snooker really ought to lend itself to a computer simulation. Despite the apparent popularity of this game, judging by the charts of best sellers (presumably this reflects the popularity of snooker itself), this is definitely not a game I want to play again, and I suspect many others would feel the same about this offering from Visions.

Title : Stranded
Supplier : Superior Software
Price : £7.95
Reviewer : Peter Lowes
Rating : *

'Stranded' is a very small and simple adventure by any standard. It took about an hour and a half to solve and it was only necessary to run through the plot twice.

The instructions consist of a brief description of the mission, which is to find your way home from an alien world on which you have been stranded. After that you are on your own.

The game runs in mode 1, giving a neat and colourful display. The top half of the screen is devoted to a picture of the current location, and the bottom half to a text window giving

the usual information in the briefest possible way. Most of the locations have a picture to go with them. However, there are far too many areas of the game, such as the maze of long twisty corridors, which are all alike and use the same picture, even if the exits are in different directions. The graphics are simple, clear and effective, but they give no clues, and remain unchanged when things happen. This means that they don't really contribute anything to the game which is a pity, considering how much memory is needed for them.

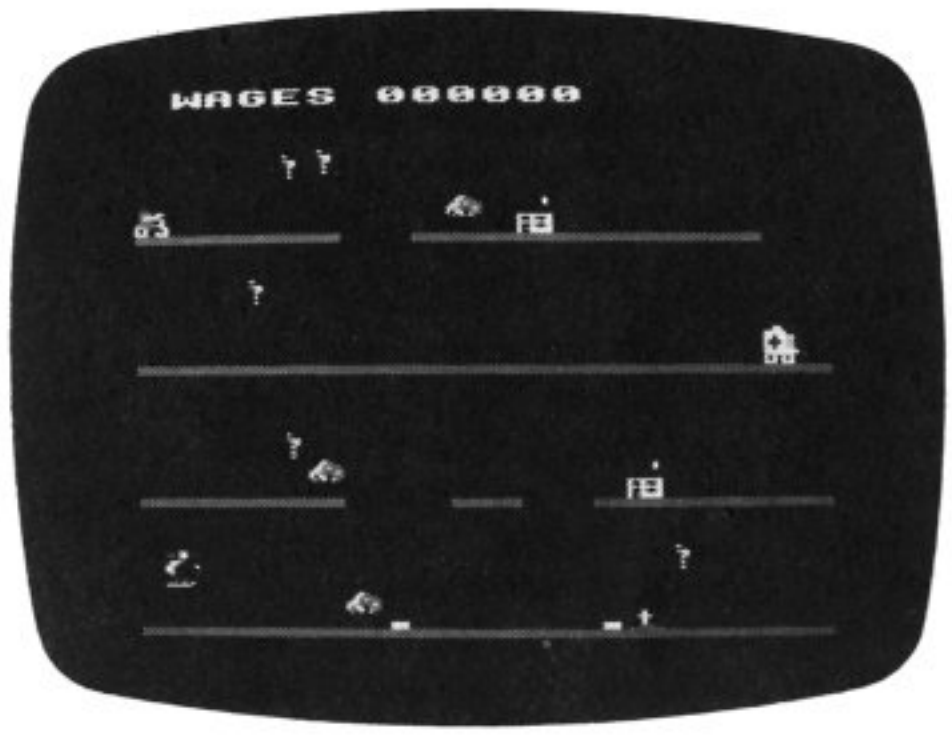
The adventure itself is pitifully small, containing only seven objects. The few so called problems are either very old or very obvious. Also the locations connect together in a very unrealistic manner. For example, exits from several locations take you to the place you you have just left. This makes the game seem bigger than it really is.

It could be argued that this game is aimed at young children or novice adventurers, but the lack of proper instructions and the very small range of commands to which the computer will respond would only frustrate a beginner. For example, you can PUSH a button but not PRESS one.

Too much has been sacrificed for the sake of the pretty pictures and not even every location has one. There are not yet many adventure games available for the Electron, as many of those written for the BBC micro used mode 7. However one comparable game is Twin Kingdom Valley from Bug Byte. This is much bigger, more logical, and has a picture to go with every location.

'Stranded' does have a game save facility which I doubt anyone will need. It records a mere nine bytes, which just about sums up the game's complexity (and appeal).

Title : Daredevil Dennis
Supplier: Visions Software
Price : £7.95
Reviewer: Alan R. Webster
Rating : ****




Dennis is a rather active stuntman, whose aim in life is to make as much money from the movies as possible. Each stunt he performs earns him big money, but there are several risks that have to be taken.

There are three different scenes. The first requires Dennis to ride a motorbike, leaping over houses and engaging in other dangerous pursuits. Among the stunts in the second scene, he has to jump a light-house with a wetbike, and in the third, his stunts are all performed on skis.

The game has smooth, fast graphics and uses over thirty multicoloured 'sprite' characters. The cassette inlay card claims that the program has an interrupt driven sound routine, but this was missing on our copy. There are also six skill levels to choose from, and each level has six intermediate levels, making thirty-six in all.

Overall, this is a very pleasant game with a lot of originality and provides quite a challenge at the harder stages.

MORE NEW ADD-ONS FOR THE ELECTRON

First Byte Computers, who have already produced a joystick interface for the Electron (reviewed in ELBUG issue 7), have now brought out a similarly packaged printer interface price £34.95 inclusive, available from W.H.Smith and all good computer dealers, or direct from First Byte Computers, 10 Main Centre, Derby.

A company called Slogger Ltd have launched a ROMBOX for the Electron at £35.00 inclusive. This unit allows you to plug in up to eight ROMs or EPROMs. Contact Slogger Ltd, 215 Beacon Road, Chatham, Kent ME5 7BU.

# DEBUGGING PROGRAMS (Part 2)

**by Mike Williams**

We conclude the two part article that we started last month, by looking at some further errors that can be introduced when typing in programs at the keyboard. Many of these potential sources of error can occur both in programs that you have typed in from magazine listings and in programs that you develop and write yourself.

This month we shall be looking at some of the errors that are likely to arise when a program uses arrays. The most obvious indication of these errors is the display of an error message with the word "array" or "subscript" in it. Many of the programs published in ELBUG make use of arrays, so we had better begin by looking at arrays and how they are used.

## DEFINING AND USING ARRAYS

An array is a collection or list of items that it is useful to treat in a common way. Just as variables can be used to store numbers or strings of characters, so also can arrays (or lists). All the items in an array must be of the same type; you cannot mix numbers and characters together. Each array is given its own unique name, and only the use of a subscript distinguishes between the different items in the array. What is a subscript you may well ask? Well it's a number which is used to differentiate between the various items by pointing to, or indicating, the position of an item in the array. Each item in an array is often referred to as an 'element' of that array.

We will look at a simple example to explain matters further. Let's assume that we have an array A( ), and that it has eleven elements Ø to 1Ø inclusive. That gives us the elements:

A(Ø), A(1), A(2), A(3), A(4), A(5)
A(6), A(7), A(8), A(9), A(1Ø)

In each case the number enclosed in the brackets is the subscript. Notice too, that in our example the first element is in position Ø (and not 1 as you might have expected). This is just typical of computing, where the first item of anything is often taken as Ø (even some computing books have numbered their first chapter Ø). You can always use an array starting at 1 if you wish, though this is a little wasteful as we shall see very shortly.

Before you can use any array in a program, you have to tell the computer how many elements it is going to have, or rather, what the maximum subscript is going to be. In our previous example, the maximum subscript is 1Ø, though the number of elements is 11. The DIM statement (short for dimension) is used for this purpose, and you simply enter a line as

```
DIM A(1Ø)
```

in order to set up an array with maximum subscript of 1Ø, before you use it. What this does is to allow the computer to set aside a block of memory for this purpose. Even if you only use 5 elements of the array in the program, the memory for all 11 will have been allocated. The computer always assumes that the first element will be the Ø one, which is why it is a little more economical to use elements Ø to 9, for example, rather than 1 to 1Ø. In most cases however, the extra memory used by starting at element 1 will be negligible and have no adverse effect on the running of the program.

Once our array has been dimensioned, with the elements going up to a maximum of 1Ø, we cannot then use a subscript larger than 1Ø (or increase the size of the array later). If you do try to use element 11 say, the computer will tell you that you can't in the only way it knows how, with an error message:

"Subscript at line....."

with an appropriate line number of course.

If the idea of arrays is new to you, then the following short program should help to clarify what we have been doing. In this program, an array called 'FRED' is defined, and the value of

each of its elements (in fact Ø) is then printed out. There is a deliberate mistake in the program though. When you type it in, see if you can spot this before continuing to read further.

```
10 DIM FRED(12)
20 FOR I%=1 TO 13
30 PRINT "Value of element ";I%;" is...";FRED(I%)
40 NEXT I%
```

The array consists of 13 elements, but numbered from Ø to 12. Thus when the FOR-NEXT loop tries to print out the value of the 13th element in line 30 it produces the error message "Subscript at line 30". Notice too that the subscript in line 30 is written as a variable (I%). Arrays are very often found in FOR-NEXT loops where we want to carry out the same process on each element (here to print the value of each one).

One small but important point to note about arrays is that the process of dimensioning an array automatically causes the value of each element to be set to zero (or to a null or empty character string if it is a string array). You do not have to assign values to arrays before you can use them, unlike the situation with ordinary variables (the "No such variable..." error message that we dealt with last month).

## CHECKING ARRAY ERRORS

If the subscript of an element is given as a variable, or even as a more complex expression, it can be quite difficult to check what is going wrong. If the "Subscript...." error message does appear, try listing first the line which the computer says is in error, and then in immediate mode print the values of any subscripts in that line. In our simple example above, this would mean listing line 30 (LIST 30) and then printing the value of I% (PRINT I%). This at least allows you to check if the subscript really is larger than it should be. Note too, that a subscript cannot be less than zero, so a negative subscript is always wrong.

Incidentally, if a subscript ever has a value which is not a whole number (or integer), then the computer just truncates the value. Thus FRED(Ø.99) would be treated as FRED(Ø). Of course this shouldn't occur (subscripts are whole numbers by definition), but it is worth knowing what will happen if it does.

In practice, the "Subscript..." error message can be caused by all sorts of errors. If it should happen to you, then check very carefully that each array in the program is correctly dimensioned, and that you haven't made a mistake in the spelling of the name of the array, either where it is dimensioned, or wherever it is used. If you do get the name of an array wrong in a dimension statement, the the error message will be "Array at line...." rather than the "Subscript..." message already described. This is because it appears to the computer that you have not dimensioned the array which you are now using.

This error can also appear rather unexpectedly on some occasions. For example, one problem that can sometimes arise is in the mis-spelling of one of the built in Basic function names. Suppose you typed SIM(angle), instead of the correct SIN(angle), using the SIN function. You would then get the "Array..." error message because the computer thinks you are trying to use an array called SIM which hasn't been dimensioned. The computer can't read your mind to see what you really meant.

## MORE PROBLEMS WITH SPACES

Last month we talked about some of the problems that can arise through the mishandling of spaces in a program. A further example of this occurs with arrays, and also with functions like SIN used above. You should NEVER leave a space between the name of an array or function and the following bracket. If this should happen by accident then the "No such variable..." error message may result.

## TWO DIMENSIONAL ARRAYS

Up to now all our references have been to what are known as "one dimensional arrays", that is arrays which we can think of as a simple list of items. However, it is just as feasible to use two dimensional arrays, and these are best thought of as tables

with both rows and columns. For example, we might have a program which uses an array dimensioned as SALARY(5,12) which is used to store monthly salaries for each of five years. The array now has two subscripts, one indicating the year and one the month. Thus the element SALARY(2,6) would be the salary for the 6th month (June) in the second year.

Although such two dimensional arrays are likely to appear more complicated when used in programs, particularly when the subscripts are not just simple numbers but are variables or expressions, the same problems are likely to arise as already described, and the advice given then is just as applicable.

Debugging programs is rather like trying to solve a mystery story or 'Who Dunnit'. The clues you need are all there if you only know where to look and what to look for. Just as with fictional tales, all is not what it seems and and you may find several red herrings on the way. Above all else, remain cool, calm and collected, and work logically and methodically. Use the power of the computer itself to help you as much as possible along the way by printing the values of variables and subscripts etc. as described above.

This completes this two part article on debugging, but it is an important topic and one which we shall no doubt return to again in due course.

# HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

ROUNDING DECIMALS - D.Gibson

To round a decimal to the nearest integer, obtain the integer of the number after adding Ø.5 to it. Thus X%=INT(X+Ø.5) turns X into the nearest integer (whereas INT(X) would simply round X down). Similarly, X%=(INT(X*1Ø+Ø.5))/1Ø rounds to the nearest decimal place, and X%=(INT(X*4+Ø.5))/4 rounds X to the nearest quarter.

MODE CHANGE BY VDU COMMAND - J.Forshaw

To change display mode from within a procedure, use the VDU22,n command where n is the number of the mode that you want. Basic will not normally allow you to use the MODE instruction within a procedure function.

CAPS LOCK POKE - G.Edwards

You can can poke to a location in memory to control the 'Caps Lock' mode of the Electron. ?&25A=&2Ø sets 'Caps Lock' on and ?&25A=&3Ø switches it off. (Don't forget the *FX2Ø2 commamd from last month's hints for a more official way to do this).

SPOOLING GRAPHICS - N.Kelly

Opening a spooled file before beginning a graphics program will save whatever is generated on to tape for playing back again another time. Type *SPOOL followed by the name of the file to be created and then run any program you wish that draws on the screen (you could include the *SPOOL instruction in the program). A file full of the characters that are sent to the screen will be created on tape (if you're using your cassette recorder). At the end of the program type *SPOOL to complete the saving operation. You can now run the graphics creation sequence through again without running the program simply by typing *EXEC followed by the file name of the file that you have just created.

# DARTBOARD

**By J. Crombie**

Have you ever wondered how top dart players such as Eric Bristow and Jocky Wilson reach the heights of perfection when they play? Well, ELBUG provides one answer, would you believe, with a program that will help you play better than ever before, and maybe score that fabulous one hundred and eighty!

Dartboard is a game for two players (though you can easily take both parts if you wish) using either the keyboard or joysticks (if you have an Acorn Plus-1 or similar joystick expansion fitted) to control your aim. If you choose to use joysticks then you may still use the keyboard for fine adjustments.

This is the traditional game of darts where each player takes turns to throw three darts to reduce his score down from the starting value of 501, finishing on a double or a 'Bulls-eye'.

You may use either a joystick or the keyboard (selected at the start of the match) to aim a dart. Then press a number key (1 to 9) to throw the dart. Select the relative strength of the throw by the size of the number, 1 being the weakest and 9 being the strongest.

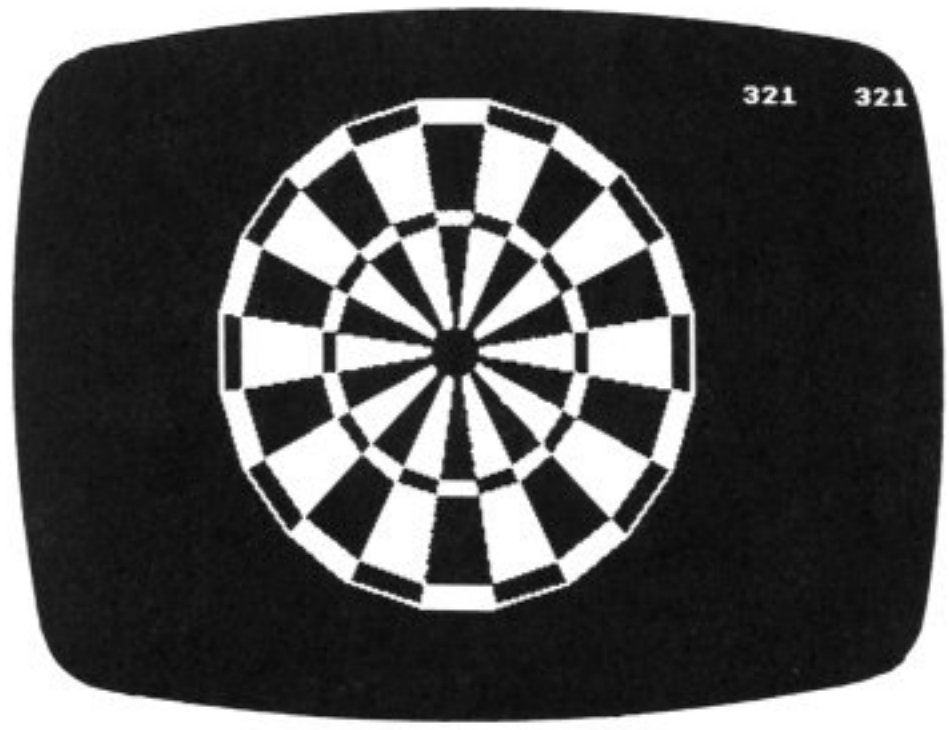
The keys for controlling direction are 'Z' and 'X' for left and right, and '*' and '?' for up and down.

As you throw your darts the computer will automatically count up the total of your three throws and display your score at the right-hand side of the screen.

When a game has finished you can choose either to play another game by pressing the spacebar or see the result of the match displayed on the screen by pressing the Return key.

PROGRAM NOTES

The program is quite well structured using a number of clearly named procedures to do most of the work. These procedures are listed below with a brief description of the function of each.




| | | |
|---|---|---|
| 1000 | PROCboard | Draws dartboard on screen. |
| 1180 | PROCsector | Draws one sector of the dartboard. |
| 1270 | PROCmovesight | Moves target point for dart using joystick or keyboard. |
| 1490 | PROCdart | Throws dart. |
| 1590 | PROCtotal | Calculates dart score. |
| 1730 | PROCremovedarts | Removes darts from board. |
| 1810 | PROCendgame | Displays match result. |
| 1910 | PROCkeyorjoy | Displays front page and selects joystick or keyboard control. |
| 2040 | PROCinit | Initialise sound envelopes, user-defined characters and variables. |

The main program loop is from line 180 to 310 which is repeated until one player is the winner.

```
   10 REM PROGRAM DARTS
   20 REM AUTHOR  J.Crombie
   30 REM VERSION E0.1
   40 REM ELBUG   AUG/SEP. 1984
   50 REM PROGRAM SUBJECT TO COPYRIGHT
   60 :
  100 ON ERROR GOTO 2080
  110 MODE1:PROCkeyorjoy
  120 CLS:VDU5:DIM SCORE%(20),SUM(2),DX
%(3),DY%(3),game%(2)
  130 PROCinit
  140 PROCboard
  150 REPEAT
  160 COLOUR0:PRINT"PLR1  PLR2 501   50
1"
  170 COLOUR3:VDU28,30,31,39,4
  180 REPEAT
  190 TIME=0:Player%=Player%+1
  200 IF Player%=3 Player%=1
  210 sum%=0:dart%=1
  220 REPEAT
  230 winner%=FALSE
  240 PROCmovesight(dart%)
  250 IF SUM(Player%)=sum% AND double%=
1 THEN winner%=TRUE:GOTO310
  260 IF SUM(Player%)-sum%<2 THEN VDU4:
PRINTTAB(6*(Player%-1)+1);"---  ";:VDU5
:PROCremovedarts(dart%):UNTIL winner%=F
ALSE:GOTO310
  270 dart%=dart%+1
  280 UNTIL dart%>3
  290 PROCremovedarts(3)
  300 SUM(Player%)=SUM(Player%)-sum%:VD
U4:PRINTTAB(6*(Player%-1)+1);SUM(Player
%);SPC(2);:VDU5
  310 UNTIL winner%
  320 VDU4:SOUND1,-15,50,10:SOUND1,-15,
100,5:SOUND1,-15,200,5:PRINT''''''" WIN
NER     Player";Player%'''''''''':game%
(Player%)=game%(Player%)+1
  330 PRINT" PRESS KEY "
  340 A=GET:IF A=13 PROCendgame:END
  350 IF dart%=4 THEN PROCremovedarts(3
 ) ELSE PROCremovedarts(dart%)
  360VDU28,30,31,39,4,12:Player%=2:SUM
(1)=501:SUM(2)=501
  370 UNTIL FALSE
  380 END
  390 :
 1000 DEF PROCboard
 1010 FORI%=1 TO 20
 1020 PROCsector(I%,3,500,500,406)
 1030 PROCsector(I%,3*(I% MOD 2),500,50
0,400)
 1040 PROCsector(I%,3*(ABS((I% MOD 2)-1
)),500,500,365)
 1050 PROCsector(I%,3*(I% MOD 2),500,50
0,225)
 1060 PROCsector(I%,3*(ABS((I% MOD 2)-1
)),500,500,200)
 1070 NEXT
 1080 FOR I%=1 TO 20:PROCsector(I%,0,50
0,500,40):PROCsector(I%,1,500,500,15):N
EXT
 1090 FOR I%=1 TO 20
 1100 MOVE450*SIN(RAD(18*I%-16))+471,440
*COS(RAD(18*I%-16))+500:READ A$:PRINTA$
 1110 GCOL0,2:MOVE430+I%,1000+I%:PRINT"
DARTS":GCOL0,1
 1120 NEXT
 1130 GCOL0,0:MOVE430+I%,1000+I%:PRINT"
DARTS"
 1140 RESTORE:FOR I%=1 TO 20:READ SCORE
%(I%):NEXT
 1150 VDU4,28,30,31,39,0,19,2,4,0,0,0:C
OLOUR130:CLS
 1160 ENDPROC
 1170 :
 1180 DEFPROCsector(N%,C%,X%,Y%,S%)
 1190 GCOL0,C%
 1200 MOVEX%,Y%
 1210 MOVE S%*SIN(RAD(18*(N%-1)-9))+X%,
 S%*COS(RAD(18*(N%-1)-9))+Y%
 1220 PLOT85,S%*SIN(RAD(18*N%-9))+X%,S%
*COS(RAD(18*N%-9))+Y%
 1230 ENDPROC
 1240 :
 1250 DATA 20,1,18,4,13,6,10,15,2,17,3,
19,7,16,8,11,14,9,12,5
 1260 :
 1270 DEFPROCmovesight(dartno%)
 1280 VDU5:F%=0
 1290 C%=A%:D%=B%
 1300 IF kb% GOTO 1350
 1310 IF ADVAL(2*(Player%-1)+1)>50000 A
%=A%-8
 1320 IF ADVAL(2*(Player%-1)+2)<8000 B%
=B%-8
 1330 IF ADVAL(2*(Player%-1)+1)<8000 A%
=A%+8
 1340 IF ADVAL(2*(Player%-1)+2)>50000 B
%=B%+8
 1350 IF INKEY-98 A%=A%-8
 1360 IF INKEY-67 A%=A%+8
 1370 IF INKEY-73 B%=B%+8
 1380 IF INKEY-105 B%=B%-8
 1390 IF A%<0 A%=0 ELSE IF A%>900 A%=900
 1400 IF B%<0 B%=0 ELSE IF B%>900 B%=900
 1410 IF TIME>300 GOTO 1440
 1420 *FX15,1
 1430 GOTO1450
 1440 KEY%=INKEY(0):IF KEY%>48 AND KEY%
<58 THEN GCOL3,2:MOVEC%,D%:VDU224:PROCd
art(A%,B%,dartno%):ENDPROC
 1450 GCOL3,2:IF F%=1 MOVEC%,D%:VDU224
 1460 F%=1:MOVEA%,B%:VDU224
 1470 GOTO1290
 1480 :
 1490 DEFPROCdart(X%,Y%,dartno)
 1500 GCOL3,1
```

```
 1510 X%=X%+16+RND(32*(KEY%-&30))-8*(KE
Y%-&30):Y%=Y%-12+RND(32*(KEY%-&30))-8*(
KEY%-&30)-12*(&3A-KEY%)
 1520 MOVEX%-20,Y%-20:DRAWX%+20,Y%+20
 1530 MOVEX%-20,Y%+20:DRAWX%+20,Y%-20
 1540 DX%(dartno)=X%:DY%(dartno)=Y%
 1550 SOUND0,-15,5,1
 1560 PROCtotal(X%,Y%)
 1570 ENDPROC
 1580 :
 1590 DEFPROCtotal(X%,Y%)
 1600 IFY%=500 Y%=499
 1610 ANGLE%=DEGATN((X%-500)/(Y%-500)):
radius%=SQR((X%-500)^2+(Y%-500)^2)
 1620 IF Y%-500<0 THEN ANGLE%=ANGLE%+18
0 ELSE IF ANGLE%<-9 THEN ANGLE%=ANGLE%+
360
 1630 SEC%=(ANGLE%+9)/18
 1640 DARTSCORE%=SCORE%(SEC%+1):double%
=0
 1650 IF radius%<406 AND radius%>365 TH
EN DARTSCORE%=DARTSCORE%*2:double%=1
 1660 IF radius%>200 AND radius%<225 TH
EN DARTSCORE%=DARTSCORE%*3
 1670 IF radius%>14 AND radius%<40 THEN
 DARTSCORE%=25
 1680 IF radius%<15 THEN DARTSCORE%=50:
double%=1
 1690 IF radius%>406 THEN DARTSCORE%=0
 1700 sum%=sum%+DARTSCORE%
 1710 ENDPROC
 1720 :
 1730 DEF PROCremovedarts(dartno%)
 1740 GCOL3,1
 1750 FORI%=1 TO dartno%
 1760 MOVEDX%(I%)-20,DY%(I%)-20:DRAWDX%
(I%)+20,DY%(I%)+20
 1770 MOVEDX%(I%)-20,DY%(I%)+20:DRAWDX%
(I%)+20,DY%(I%)-20
 1780 NEXT
 1790 ENDPROC
 1800 :
 1810 DEFPROCendgame
 1820 VDU22,1
 1830 COLOUR1:PRINTTAB(15,4);"DARTS"
 1840 SOUND1,-15,10,5:SOUND1,-15,50,5:S
OUND1,-15,25,10
 1850 COLOUR2:PRINTTAB(5,12);"Player 1
has won ";game%(1);" games"
 1860 COLOUR2:PRINTTAB(5,14);"Player 2
has won ";game%(2);" games"
 1870 COLOUR1:IF game%(1)=game%(2) THEN
 PRINTTAB(1,20);"I therefore declare th
e contest a draw"'':ENDPROC
 1880 PRINTTAB(1,20);"I therefore declar
e the winner ";:IF game%(1)>game%(2) THE
N PRINT"Player 1" ELSE PRINT"Player 2"
 1890 ENDPROC
 1900 :
 1910 DEFPROCkeyorjoy
 1920 VDU23;1,0;0;0;0;
 1930 COLOUR1:PRINTTAB(12,3)"Dartboard"
 1940 COLOUR2:PRINTTAB(11,6)"by J.Cromb
ie"
 1950 COLOUR3:PRINTTAB(9,12)"Are you pl
aying"''TAB(10)"from Keyboard"
 1960 PRINTTAB(12)"or Joystick"
 1970 REPEAT:VDU7
 1980 COLOUR2:PRINTTAB(15,17)"?";
 1990 G$=GET$:PRINTTAB(19,17)G$
 2000 UNTIL G$="K" OR G$="J"
 2010 IF G$="K" kb%=TRUE ELSE kb%=FALSE
 2020 ENDPROC
 2030 :
 2040 DEFPROCinit
 2050 VDU23,224,8,8,8,255,8,8,8,8:A%=50
0:B%=500:SUM(1)=501:SUM(2)=501:@%=0:Pla
yer%=2:game%(1)=0:game%(2)=0
 2060 ENDPROC
 2070 :
 2080 ON ERROR OFF
 2090 MODE 6
 2100 IF ERR=17 END
 2110 REPORT:PRINT" at line ";ERL
 2120 END
```

# POINTS ARISING

SOUND ENVELOPE EDITOR

In the update to the Sound Envelope Editor that we published in the June issue of ELBUG, an inexplicable error crept in. The NEXTF% contained in line 360 is quite redundant and should be omitted if you use this correction.

737 FLIGHT SIMULATOR

In our review of the 737 Flight Simulator by Salamander Software we obviously got carried away by the excellence of the display when we referred to the accompanying engine noise. In fact, the Electron version of this simulation does not have this feature. Our apologies if we inadvertantly misled anyone over this.

Subscriptions and Software Help Line
St Albans (0727) 60263
Manned Mon-Fri 1pm-4pm.

## BACK ISSUES AND SUBSCRIPTIONS

### BACK ISSUES (Members only)

All back issues will be kept in print (from November 1983). Send 90p per issue PLUS an A5 SAE to the subscriptions address. Back copies of BEEBUG are available to ELBUG members at this same price. This offer is for members only, so it is ESSENTIAL to quote your membership number with your order. Please note that the advertising supplements are not supplied with back issues.

### Subscription and Software Address

ELBUG
PO BOX 109
High Wycombe
Bucks

### SUBSCRIPTIONS

Send all applications for membership, and subscription queries to the subscriptions address.

MEMBERSHIP COSTS:

U.K.
£5.90 for 6 months (5 issues)
£9.90 for 1 year (10 issues)
Eire and Europe
Membership £16 for one year.
Middle East £19
Americas and Africa £21
Elsewhere £23
Payments in Sterling preferred.

### SOFTWARE (Members only)

This is available from the software address.

## MAGAZINE CONTRIBUTIONS AND TECHNICAL QUERIES

Please send all contributions and technical queries to the editorial address opposite. All contributions published in the magazine will be paid for at the rate of £25 per page.

We will also pay £10 for the best Hint or Tip that we publish, and £5 to the next best. Please send all editorial material to the editorial address opposite. If you require a reply it is essential to quote your membership number and enclose an SAE.

### Editorial Address

ELBUG
PO Box 50
St Albans
Herts

ELBUG MAGAZINE is produced by BEEBUG Publications Ltd.
Editor: Mike Williams.
Production Editor: Phyllida Vanstone.
Technical Assistants David Fell, Nigel Harris and Alan Webster.
Managing Editor: Lee Calcraft.
Thanks are due to Sheridan Williams, and Adrian Calcraft for assistance with this issue.

# New Elbug Binders

We have produced an attractive hard-backed binder for the ELBUG magazine. These binders are green in colour with "ELBUG" in gold lettering on the spine and allow for the whole of one volume of the magazine to be stored as a single reference book.

Each binder will accommodate 10 ELBUG magazines, and is supplied with 12 wires to enable the index and the latest copy of the supplement to be included within the binder if required. Individual issues may be easily added and removed, allowing for the latest volume to be filed as it arrives.

The price of the new ELBUG binder is £3.90 including VAT, please add 50p post and packing for delivery within the U.K. Overseas members please send the same amount, this will cover the extra postage but not VAT.
Plese send to:
BEEBUGOFT, PO BOX 109, High Wycombe, Bucks, HP10 8HQ

## THE BEST OF ELBUG ON CASSETTE

Many of the best programs published in ELBUG have been collected together and published by Penguin Books under the name "Games and other programs for the Acorn Electron" at £3.95. This book is part of the Penguin Acorn Computer Library and at present there is just one other title available though others are planned.

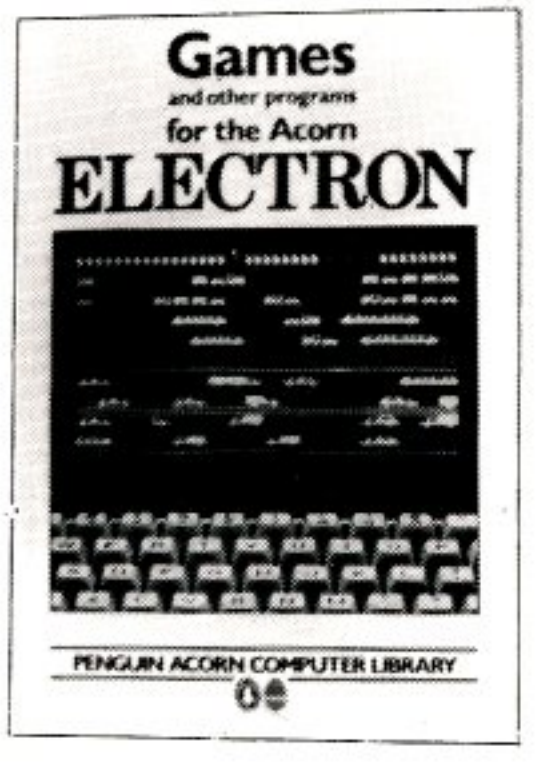
There are 20 programs in all in four different categories:

Action Games
- Munch-Man
- Mars Lander
- Invasion
- Robot Attack
- Hedgehog

Thought games
- Higher/Lower
- Five-Dice
- Life
- Anagrams
- Return of the Diamond

Visual Displays
- Union Jack
- Square Dance
- Ellipto
- Screenplay
- 3-D Rotation

Utilities
- Sound Wizard
- Bad Program Lister
- 3-D Lettering
- Bad Program Rescue
- Double Height Text




All 20 programs are now available on cassette from our software address (in High Wycombe) price £7 to members and £9 to non-members, plus 50p post & packing in either case.

# ELBUG MAGAZINE CASSETTE

To save wear and tear on fingers and brain, we offer, each month, a cassette of the programs featured in the latest edition of ELBUG. The first program on each tape is a menu program, detailing the tape's contents, and allowing the selection of individual programs. The tapes are produced to a high technical standard by the process used for the BEEBUGSOFT range of titles.

Magazine cassettes have been produced for each issue of ELBUG from Volume 1 Number 1 onwards and are all available from stock, priced £3.00 each inclusive of VAT. See below for ordering information.

This months cassette includes:

This month's cassette (Vol. 1 No. 9) includes:
Codebreaker game, Harmonograph display, a Mini Text Editor for simple word processing, Build a House graphics display, Kayak Kapers game, graphics demonstration program – GCOLMANIA, Dartboard game and a program to generate a TV/monitor Test Card.

## MAGAZINE CASSETTE SUBSCRIPTION

We are also able to offer ELBUG members subscription to the magazine cassette, this gives the added advantage of receiving the cassette at around the same time as the magazine each month. Subscriptions may either be for a period of 1 year or 6 months. (NOTE Magazine cassettes are produced 10 times each year).

If required, subscriptions may be backdated as far as Volume 1 Number 1, so when applying please write to the address below quoting your membership number and the issue from which you would like your subscription to start.

## MAGAZINE CASSETTE ORDERING INFORMATION

Individual ELBUG Magazine Cassettes £3.00.
P & P: Please add 50p for the first and 30p for each subsequent cassette.
Overseas orders: Please send the same amount, this will include the extra post but not VAT.

Magazine Cassette Subscription

| | | | | |
|---|---|---|---|---|
| 1 YEAR (10 issues) | £33.00 Inclusive..... | O'SEAS | £39.00 | No VAT payable |
| 6 MONTHS (5 issues) | £17.00 Inclusive..... | O'SEAS | £20.00 | No VAT payable |

Please be sure to specify that you require subscription to the ELBUG magazine cassette (as opposed to the BEEBUG cassette), and enclose your membership number with a cheque made payable to BEEBUGSOFT.

Please send to..

**ELBUG Magazine Cassette, BEEBUGSOFT, PO Box 109, High Wycombe, HP10 8HQ**

Printed in England by Staples Printers St Albans Limited at The Priory Press.